$(\mathcal{C}+)^{++}$: An Action Language for Modelling Norms and Institutions

Marek Sergot

Department of Computing
Imperial College, London
mjs@doc.ic.ac.uk

Technical Report 2004/8
June 2004; revised April 2005

Abstract

The language $\mathcal{C}+$ of Giunchiglia, Lee, Lifschitz, McCain, and Turner (2004) is a formalism for specifying and reasoning about the effects of actions and the persistence ('inertia') of facts over time. An 'action description' in $\mathcal{C}+$ is a set of $\mathcal{C}+$ laws which define a labelled transition system of a certain kind. This document presents $(\mathcal{C}+)^{++}$, an extended form of $\mathcal{C}+$ designed for representing norms of behaviour and institutional aspects of (human or computer) societies. There are two main extensions. The first is a means of expressing 'counts as' relations between actions, also referred to as 'conventional generation' of actions. The second is a way of specifying the permitted (acceptable, legal) states of a transition system and its permitted (acceptable, legal) transitions.

1 Introduction

The language $\mathcal{C}+$ of Giunchiglia, Lee, Lifschitz, McCain, and Turner (2004) is a formalism for specifying and reasoning about the effects of actions and the persistence ('inertia') of facts over time. An 'action description' in $\mathcal{C}+$ is a set of $\mathcal{C}+$ laws which define a labelled transition system of a certain kind. Implementations supporting a wide range of querying and planning tasks are available, notably in the form of the 'Causal Calculator' CCalc[1]. This document presents $(\mathcal{C}+)^{++}$, an extended form of $\mathcal{C}+$ designed for representing norms of behaviour and institutional aspects of (human or computer) societies. There are two main extensions. The first is a means of expressing 'counts as' relations between actions, also referred to as 'conventional generation' of actions. The second is a way of specifying the permitted (acceptable, legal) states of a transition system and its permitted (acceptable, legal) transitions.

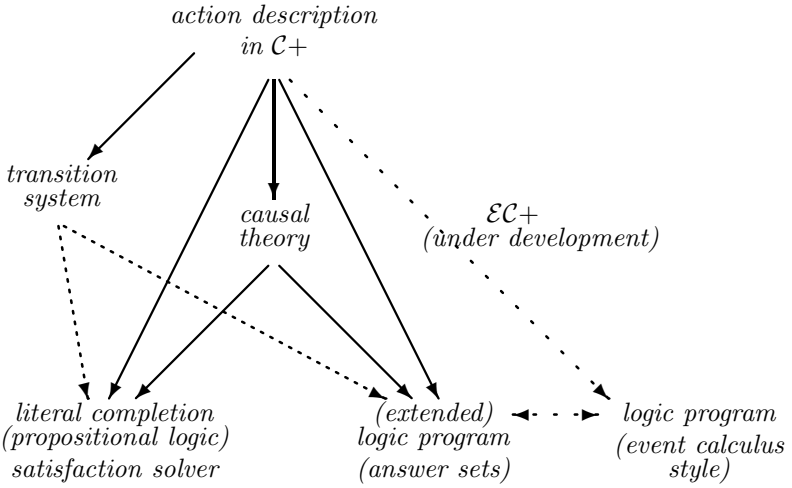
There are two aims: to investigate how far one can get by building on transition systems, and to determine how far one can get by building on the language $\mathcal{C}+$ in particular.

The language $\mathcal{C}$ was introduced by Giunchiglia and Lifschitz (1998). It applies the ideas of 'causal theories' (Lifschitz, 1997; Turner, 1999) to reasoning about the effects of actions and the persistence ('inertia') of facts ('fluents'), building on earlier suggestions by McCain and Turner (1997). A summary of $\mathcal{C}$ is included

[1] `http://www.cs.utexas.edu/users/tag/cc`

in the survey article by Gelfond and Lifschitz (1998). (Giunchiglia and Lifschitz, 1999) discusses relationships to situation calculus. $\mathcal{C}+$ extends $\mathcal{C}$ by allowing multi-valued fluents as well as Boolean fluents (Giunchiglia et al., 2001) and generalises the form of rules in the language in various ways. The definitive presentation of of $\mathcal{C}+$, and of the underlying formalism of 'nonmonotonic causal theories', is provided in (Giunchiglia et al., 2004). $\mathcal{C}+$ and the Causal Calculator CCALC have been applied successfully to a number of non-trivial benchmark examples in the temporal reasoning literature (see e.g. (Akman et al., 2004) and the CCALC website). We have used it in our own work to construct executable specifications of agent societies (see e.g. (Artikis et al., 2003a,b)).

The language $\mathcal{C}+$ provides a means of constructing a *transition system* with certain properties. A separate language is used for making assertions about this transition system—about what is true at various states in runs of the transition system—and for expressing queries about them. One implementation route is via the translation of a $\mathcal{C}+$ action description into a causal theory, and thence into a set of formulas of (classical) propositional logic (its 'literal completion'). This is the method used by CCALC: it performs the required translations and then invokes a standard propositional satisfaction solver to find models of the $\mathcal{C}+$ action description. An alternative implementation route is provided by translations into extended logic programs (Lifschitz and Turner, 1999). We are developing another translation into logic programs with a different computational behaviour ($\mathcal{EC}+$ in the diagram). That line of development is not covered in this document. See (Craven and Sergot, 2003).



In what follows, we give particular emphasis to the transition system semantics. We want to exploit the bridge this provides to other standard methods and tools in logic and computer science, such as temporal logics and model checking techniques.

Section 2 provides an overview of the two language extensions by way of motivation. Sections 3–7 present the language $\mathcal{C}+$ and the formalism of nonmonotonic causal theories, and the transition system semantics of $\mathcal{C}+$ on which the rest of the development builds. Section 8 presents the language $(\mathcal{C}+)^+$ and the

treatment of 'counts as'. Section 9 presents the language $(\mathcal{C}+)^{++}$ and its treatment of permission. Section 10 summarises and gives a brief outline of further developments.

2 Outline

We want to be able to represent and reason about procedures and protocols—what actions are possible, permitted, valid at each stage; what rights, powers, obligations are created and destroyed as the protocol/procedure progresses; what kind of normative or institutional relations are created when the protocol/procedure reaches completion. We want to be able to determine from a recorded narrative of actions that have been performed what actions are now possible, permitted, valid, and what their effects will be; we should also like to be able to prove general properties of these protocols (termination, 'fairness', that such and such a protocol has a particular property, that two protocols have identical effects, and so on). Typical examples include auction protocols, rules of procedure, argumentation/dialogue games, negotiation protocols, contract formation, business rules and procedures, procedural law.

We have taken (labelled) transition systems as the basic representational structure. So we will have fluents (state variables, Boolean and non-Boolean, whose values vary from state to state), transition labels to represent actions, and in $\mathcal{C}+$, a way of specifying the effects of actions on fluents. $\mathcal{C}+$ expressions of the form α causes F if G express that F holds after any transition of type α when G holds in the state immediately preceding the transition. Other language features of $\mathcal{C}+$ will be presented later. $\mathcal{C}+$ provides a treatment of default persistence ('inertia') of fluents, indirect effects of actions, and a form of concurrent execution of actions. It also allows us to express properties of transitions as a whole, such as saying that a transition is 'permitted' or 'acceptable' or 'unusual'. In future developments we plan to introduce explicit representations of agents (including names for collective agents as well as for individuals), roles, and multiple institutions. These are not included yet.

The first additional feature of the extended language $(\mathcal{C}+)^{++}$ is a relationship ('counts as') between actions: we want to be able to say that, in circumstances G, a transition of type α counts as a transition of type β. The effects of such a transition are those that it has by virtue of being a transition of type α and those it has by virtue of being a transition of type β. This is essentially the same idea as the 'counts as' conditional presented in (Jones and Sergot, 1996) but now transposed to a different treatment of action on a quite different semantical structure.

For example: suppose we wish to represent that birth of a person X initiates (or 'causes') that X is a citizen of the Republic (it does not matter which Republic—let it be the Republic of Great Britain) if either of X's parents is a citizen of the Republic at the time of X's birth. X's birth, moreover, makes the parents of X happy. Let $\mathit{birth}\,(X, F, M)$ and $\mathit{acquires\_cit}\,(X)$ be names of actions, and $\mathit{citizen}\,(X)$, $\mathit{parent}\,(X, Y)$, and $\mathit{happy}\,(X)$ be fluents, for X, Y, F and M ranging over some given set of person names. The expression $\mathit{birth}\,(X, F, M)$ is to be read as representing the birth of a child X to a father F and a mother M. In

the language $\mathcal{C}+$ we could write:

$$\begin{array}{l} \mathit{birth}\,(X, F, M)\ \mathsf{causes}\ (\mathit{parent}\,(X, F) \wedge \mathit{parent}\,(X, M)) \\ \mathit{birth}\,(X, F, M)\ \mathsf{causes}\ (\mathit{happy}\,(F) \wedge \mathit{happy}\,(M)) \\ \mathit{birth}\,(X, F, M)\ \mathsf{causes}\ \mathit{citizen}\,(X)\ \mathsf{if}\ (\mathit{citizen}\,(F) \vee \mathit{citizen}\,(M)) \end{array}$$

In the extended language $(\mathcal{C}+)^{++}$ the same example could be represented instead as follows:

$$\begin{array}{l} \mathit{birth}\,(X, F, M)\ \mathsf{causes}\ (\mathit{parent}\,(X, F) \wedge \mathit{parent}\,(X, M)) \\ \mathit{birth}\,(X, F, M)\ \mathsf{causes}\ (\mathit{happy}\,(F) \wedge \mathit{happy}\,(M)) \\ \mathit{birth}\,(X, F, M)\ \mathsf{counts\_as}\ \mathit{acquires\_cit}\,(X)\ \mathsf{if}\ (\mathit{citizen}\,(F) \vee \mathit{citizen}\,(M)) \\ \mathit{acquires\_cit}\,(X)\ \mathsf{causes}\ \mathit{citizen}\,(X) \end{array}$$

This $(\mathcal{C}+)^{++}$ representation separates what some (e.g. (Searle, 1969)) have called a 'brute fact', the birth of X, from its institutional effects, the acquisition of citizenship by X, as recognised by the Republic. It separates the effects that the birth of X has by virtue of being a birth (the parents of X become happy, a brute fact), and possible effects it might have by virtue of being an acquisition of citizenship (X becomes a citizen, an institutional fact). This separation is essential, we believe, to give structure to the representation, though the benefits of that are difficult to demonstrate with such a small example. Larger and more realistic examples will be presented in due course. Furthermore, it allows us to specify with much greater precision what is permitted and what is not. For suppose that the Republic forbids the birth of X when the parents of X already have another child. In the language $(\mathcal{C}+)^{++}$, this can be expressed

$$\mathsf{not\text{-}permitted}\quad \mathit{birth}\,(X, F, M)\ \mathsf{if}\quad \mathit{parent}\,(Y, F) \wedge \mathit{parent}\,(Y, M) \wedge X \neq Y$$

But does this imply that the acquisition of citizenship of X is also forbidden (not permitted) in these circumstances? No—there may be other means, such as naturalization, by which a person X can acquire citizenship, and those other means are not necessarily forbidden.

On the other hand, suppose that the children of Adam and Eve are not permitted to be citizens (for some reason or another). In the language $(\mathcal{C}+)^{++}$, we can write

$$\mathsf{not\text{-}permitted}\quad \mathit{citizen}\,(X) \wedge \mathit{parent}\,(X, \mathit{Adam}) \wedge \mathit{parent}\,(X, \mathit{Eve})$$

In $(\mathcal{C}+)^{++}$ it can be inferred from this (see Section 9), not only that $\mathit{acquires\_cit}\,(X)$ is not permitted for any child X of Adam and Eve, but also that $\mathit{birth}\,(X, \mathit{Adam}, \mathit{Eve})$ is not permitted whenever $\mathit{citizen}\,(\mathit{Adam})$ or $\mathit{citizen}\,(\mathit{Eve})$ holds (because then $\mathit{citizen}\,(X)$ holds, and $\mathit{citizen}\,(X)$ is not permitted for this X).

For the representation of protocols and procedures (such as the rules of an auction, or the formation of a contract) the 'counts as' relation provides a means—again, essential we believe—for expressing the distinction between unsuccessful attempts to perform a certain, institutionally significant, kind of action, and successful performance of that action. For example, in an offer-acceptance protocol as usually found in contract formation, we often say that X 'made an offer' to Y that such-and-such should be done even when the offer that X made was

in fact an invalid offer that has no possible effect. For example, when Jeremy aged 10 offers to sell his father's car JYU-927W to Alex we still say Jeremy 'made an offer' to Alex even though clearly Jeremy, as a 10 year old child, has no power to offer to sell the car to anyone. Jeremy 'makes an offer' to Alex only in the sense that he performs some kind of communicative act that would have been regarded as a real, valid offer had Jeremy been empowered to offer to sell the car at the time. Since Jeremy was not so empowered, he performed the communicative act (he 'made an offer') but this act was not effective (he did not 'make an offer' in the other sense of that term).

In line with the account given in (Jones and Sergot, 1996), we will distinguish between the action that attempts, possibly unsuccessfully, to produce some institutionally significant fact (such as offering to sell the car) and the action that (by definition) successfully does produce that institutionally significant fact. The notation $X{:}\alpha$ will represent that agent X performed (successfully) the action α. The notation X *signals* α will represent that agent X made an attempt, possibly successful, possibly not, to perform the action α. (The keyword 'attempts' or 'indicates' could have been chosen in place of 'signals'. It is difficult to find a suitable term that works in all contexts.) How does X 'signal' or 'indicate' his attempt to offer? There are further rules of the 'counts as' type to specify that.

So we will have rules specifying when an attempted act is successful, i.e., rules of the form:

$$(X \textit{ signals } \alpha) \ \mathsf{counts\_as} \ (X{:}\alpha) \ \mathsf{if} \ \ \ldots \textit{conditions}$$

For convenience, we shall usually specify such rules using the abbreviation pow:

$$\mathsf{pow}(X, \alpha) =_{\mathrm{def}} (X \textit{ signals } \alpha) \ \mathsf{counts\_as} \ \ (X{:}\alpha)$$

Example:

$$\mathsf{pow}(X, \textit{sell}\,(Y)) \ \mathsf{if} \ \textit{owner}\,(X, Y)$$

(The example is for illustration. It does not pretend to be a representation of the concept of selling or of ownership.)

Here pow represents what Jones and Sergot (1996) termed 'institutionalised power', a generalisation to organisational structures in general ('institutions') of the concept that in legal theory is variously called 'legal power', 'legal competence', or 'legal capacity'. Thus, in the legal context, $\mathsf{pow}(X, \alpha)$ represents that X has the legal competence/capacity to perform action α.

Permission to perform certain kinds of acts is specified separately. For example, Jeremy's father might want to tell his son that he (Jeremy) is not permitted to offer to sell car JYU-927W to anyone. He means by this:

$$\mathsf{not\text{-}permitted} \ \ \textit{Jeremy signals offer}\,(\textit{Jeremy}, Y, \textit{JYU-927W})$$

The fact that Jeremy, as a 10 year old child, is not empowered (not legally competent) to make an offer to sell the car is irrelevant—it is *attempts* by Jeremy to offer to sell the car that are not permitted.

Or to take another example, we might wish to say that in general, in a certain institution, acts of communicating offers without the requisite power/competence are regarded as anti-social and are not permitted:

$$\mathsf{not\text{-}permitted} \ \ X \textit{ signals offer}\,(X, Y, P) \ \mathsf{if} \ \ \neg\mathsf{pow}(X, \textit{offer}\,(X, Y, P))$$

In practice, we would probably want to say that making offers which are *prima facie* invalid is not permitted. That refinement is easily made.

As an *implementation* issue, we might wish to go on to specify that a system implementing a certain protocol/procedure should disable non-permitted actions. That is a separate consideration. The point is that we distinguish and specify separately:

- when attempts to perform actions 'count as' successful—these are the constitutive rules of the protocol;
- which kinds of actions (successful, unsuccessful) are permitted—these are normative-regulative rules that further classify runs of the protocol into permitted (acceptable, desirable) and not permitted (unacceptable, undesirable);
- how a system to implement/control/enforce the protocol should deal with prohibited (unacceptable, undesirable) runs of the protocols.

Examples of specifications of protocols and procedures from the area of multi-agent systems in this style, though not using the language $(\mathcal{C}+)^{++}$, may be found in (Artikis et al., 2002, 2003a,b). The aim of this report is to present the language $(\mathcal{C}+)^{++}$ and its features.

3 Preliminaries

The earliest presentations of $\mathcal{C}+$ gave the semantics of the language both in terms of transition systems and as translations to 'causal theories'. The most recent presentation (Giunchiglia et al., 2004) downplays the transition systems aspect, presenting the language as a higher-level notation for causal theories of a certain kind with the relationship to transition systems presented as a secondary issue. In our case, however, one of the main aims is to explore the suitability of (labelled) transition systems as the semantic basis for a language to represent actions and norms, and for this reason we want to emphasise the transition systems aspects. The formal semantics of $\mathcal{C}+$ and its extensions $(\mathcal{C}+)^{+}$ and $(\mathcal{C}+)^{++}$ will be given here in terms of transition systems, with translations to causal theories treated primarily as an implementation route.

The material in this section is quite standard, though the transition systems defined by $\mathcal{C}+$ do have a novel feature in that they allow formulas to be evaluated on transition labels as well as states. The main purpose of the section is to introduce terminology and some notation.

3.1 Multi-valued signatures

A *multi-valued propositional signature* σ (Giunchiglia et al., 2001, 2004) is a set of symbols called *constants*. For each constant c in σ, there is a non-empty set $dom(c)$ of values called the *domain* of c. An *atom* of a signature σ is an expression of the form $c = v$ where c is a constant in σ and $v \in dom(c)$. We will write $Atoms(\sigma)$ for the set of atoms of signature σ. An atom $c = v$ is *trivial* if the domain of c is a singleton, i.e., if $dom(c) = \{v\}$ for some value v. We write $Trivial(\sigma)$ for the set (possibly empty) of the trivial atoms of σ.

A *formula* φ of signature σ is any propositional compound of atoms of σ.

$$\varphi ::= \bot \mid \top \mid \text{any atom } c=v \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi.$$

We write $\varphi \rightarrow \psi$ for truth functional ('material') implication ($\varphi \rightarrow \psi =_{\text{def}} \neg\varphi \vee \psi$). $c \neq v$ is shorthand for $\neg(c=v)$. $\top$ (true) and $\bot$ (false) are 0-ary connectives (rather than constants with a fixed interpretation). $Form(\sigma)$ denotes the set of formulas of signature σ.

A *Boolean constant* is one whose domain is the set of truth values $\{\mathtt{t}, \mathtt{f}\}$. If p is a Boolean constant, p is shorthand for the atom $p=\mathtt{t}$ and $\neg p$ for the atom $p=\mathtt{f}$. Notice that, as defined here, $\neg p$ is an *atom* when p is a Boolean constant.

An *interpretation* of σ is a function that maps every constant in σ to an element of its domain. An interpretation I *satisfies* an atom $c=v$, written $I \models_\sigma c=v$, if $I(c) = v$. We will often omit the subscript σ when context allows. The satisfaction relation $\models_\sigma$ is extended from atoms to formulas in accordance with the standard truth tables for the propositional connectives. When X is a set of formulas we also write $I \models_\sigma X$ to signify that $I \models_\sigma \varphi$ for all formulas $\varphi \in X$. I is then a *model* for the set of formulas X. A set of formulas X is *σ-satisfiable* if it has a model, i.e., if $I \models_\sigma X$ for some interpretation I of σ. Where σ is a signature, we will write $\mathrm{I}(\sigma)$ for the set of interpretations of σ.

An expression of the form $c = d$, where both c and d are constants, is understood as shorthand for the disjunction $(c=v_1 \wedge d=v_1) \vee \cdots \vee (c=v_k \wedge d=v_k)$ where $v_1, \ldots, v_k$ are the elements of $dom(c) \cap dom(d)$. $c \neq d$ is shorthand for $\neg(c = d)$.

We write $\models_\sigma \varphi$ to mean that $I \models_\sigma \varphi$ for all interpretations I of σ. Where X is a set of formulas of signature σ, $X \models_\sigma \varphi$ denotes that $I \models_\sigma \varphi$ for all interpretations I of σ such that $I \models_\sigma X$. When X' is a set of formulas of signature σ, $X \models_\sigma X'$ is shorthand for $X \models_\sigma \varphi$ for all formulas $\varphi \in X'$. It is easy to check that (for finite X and X') $X \models_\sigma \varphi$ iff $\models_\sigma \bigwedge(X) \rightarrow \varphi$, and $X \models_\sigma X'$ iff $\models_\sigma \bigwedge(X) \rightarrow \bigwedge(X')$. $X \equiv_\sigma X'$ means that $X \models_\sigma X'$ and $X' \models_\sigma X$.

Proposition 1 *Let X and X' be sets of atoms of signature σ. Let $Trivial(\sigma)$ denote the set (possibly empty) of trivial atoms of σ. If X is σ-satisfiable, then $X \models_\sigma X'$ iff $X' \subseteq X \cup Trivial(\sigma)$.*

Proof. The notation $X \models_\sigma X'$ is shorthand for $X \models_\sigma \varphi$ for all $\varphi \in X'$, so it is sufficient to show that the result holds for all atoms of σ: we show that if X is σ-satisfiable, then $X \models_\sigma c=v$ iff $c=v \in X \cup Trivial(\sigma)$.
Left-to-right: Suppose X is σ-satisfiable and $c=v$ is an atom of σ. Suppose $c=v$ is not trivial. We show that if $c=v \notin X$ then $X \not\models_\sigma c=v$. Suppose $c=v \notin X$. There are two cases: either (i) $c=v' \in X$ for some $v \neq v'$, or (ii) there is no c atom in X. For case (i), $c=v' \in X$ implies $X = X \cup \{c=v'\}$, and since X is σ-satisfiable, so is $X \cup \{c=v'\}$. For case (ii), choose any interpretation I of σ which has $I \models_\sigma X$ and $I(c) = v'$ for some $v' \neq v$. There must be such a $v' \in dom(c)$ because, by assumption, $c=v$ is not trivial. So in both cases (i) and (ii), there is an interpretation I of σ such that $I \models_\sigma X \cup \{c=v'\}$ for some $v' \neq v$. So now we have $I \models_\sigma X$ but $I \not\models_\sigma c=v$, and so $X \not\models_\sigma c=v$.
Right-to-left: Suppose $c=v \in X \cup Trivial(\sigma)$ and $I \models_\sigma X$. We show $I \models_\sigma c=v$. If $c=v$ is trivial then $I \models_\sigma c=v$ since clearly $I \models_\sigma Trivial(\sigma)$ for any interpretation I of σ. If $c=v$ is not trivial then $c=v \in X$ and hence $I \models_\sigma c=v$ because $I \models_\sigma X$. □

The σ-satisfiability condition is required in Proposition 1. For if X is not σ-satisfiable, then $X \models_\sigma X'$ for any set $X' \subseteq \mathit{Atoms}(\sigma)$. And suppose $c=v$ is a trivial atom of a signature σ. Then $X \models_\sigma c=v$ for all $X \subseteq \mathit{Atoms}(\sigma)$, even when $c=v \notin X$.

Reduction to Boolean signatures

It is easily seen that when the domain of every constant c of the signature is finite, all multi-valued constants and interpretations can be translated to Boolean constants and standard propositional interpretations (Giunchiglia et al., 2004): replace c by a family of Boolean constants, one for each value of $\mathit{dom}(c)$, together with the constraint that exactly one of these new constants must have value `t` in any interpretation.

In other words, given a signature σ, let σ_{B} be the signature which is identical to σ for all Boolean constants of σ, and in which there is a Boolean constant of the form $c=v$ for every non-Boolean atom $c=v$ of σ. When I is an interpretation of σ, let I_{B} be the interpretation of σ_{B} such that $I_{\mathrm{B}}(p) = I(p)$ for all Boolean constants p of σ, and $I_{\mathrm{B}}(c=v) = \mathtt{t}$ iff $I(c) = v$ for all non-Boolean atoms $c=v$ of σ.

An atom $c=v$ can be viewed as a classical, propositional atom. Finding a model of a set X of formulas of a (multi-valued) signature σ can be reduced to finding a classical model of X together with the following set of additional formulas $\mathit{Boolean}(\sigma)$:

$$\bigvee_v (c=v) \;\wedge\; \bigwedge_{v \neq w} \neg(c=v \wedge c=w) \qquad \text{for all } c \in \sigma$$

That is, $I \models_\sigma \varphi$ iff $I_{\mathrm{B}} \models_{\sigma_{\mathrm{B}}} \{\varphi\} \cup \mathit{Boolean}(\sigma)$, and hence

$$X \models_\sigma \varphi \quad \text{iff} \quad X \cup \mathit{Boolean}(\sigma) \models_{\mathrm{PL}} \varphi$$

where $\models_{\mathrm{PL}}$ denotes classical, truth-functional entailment.

3.2 Transition systems

A *labelled transition system* is a structure $\langle S, \mathbf{A}, R\rangle$ in which

- S is a (non-empty) set of 'states';
- $\mathbf{A}$ is a (non-empty) set of transition labels (also called 'events');
- R is a set of transitions, $R \subseteq S \times \mathbf{A} \times S$.

It does not matter whether we think of the labelled transitions as a single three-place relation R, as here, or as a family of binary relations $\{R_\varepsilon\}_{\varepsilon \in \mathbf{A}}$. The former is chosen here for consistency with published accounts of the language $\mathcal{C}+$. A transition system can be depicted as a labelled directed graph. Every state s is a node of the graph. Labelled directed edges of the graph are the tuples (s, ε, s') of R.

We are free to interpret the labels on the transitions in various ways. The usual way is to see each label as corresponding to execution of an action or perhaps several actions concurrently. It is then usual to call the transition label an 'event'. The triple (s, ε, s') represents execution of event ε in state s leading

(possibly non-deterministically) to the state s'. An event ε is *executable* in s when there is at least one tuple (s, ε, s') in R. An event ε is *deterministic* in s if there is at most one such s'.

Paths, 'runs', or 'histories' A *run* or *trace* of a transition system is a finite or infinite (ω length) path through the system. (One or other of the terms *run* or *trace* is often reserved to refer to infinite length paths. We will use 'run' and 'path' interchangeably and avoid the use of the term 'trace'. The account of $\mathcal{C}+$ in (Giunchiglia et al., 2004) uses the term 'history'.)

Let $\langle S, \mathbf{A}, R\rangle$ be a transition system. A *run* (or *path* or *history*) of length m is a sequence

$$s_0\, \varepsilon_0\, s_1\, \cdots\, s_{m-1}\, \varepsilon_{m-1}\, s_m \qquad (m \geq 0)$$

such that $s_0, s_1, \ldots, s_m \in S$, $\varepsilon_0, \ldots, \varepsilon_{m-1} \in \mathbf{A}$, and $(s_i, \varepsilon_i, s_{i+1}) \in R$ for $0 \leq i < m$.

Sometimes there is a distinguished set $S_0 \subseteq S$ of *initial states*. All runs (or histories) are then defined so that their first state $s_0 \in S_0$. If there is a single initial state $S_0 = \{s_0\}$ then the set of all runs of the transition system can be seen as a *tree* rooted in s_0.

3.3 Query languages

A wide variety of languages—we will call them query languages—can be interpreted on labelled transition systems. These include simple propositional languages, as well as temporal logics such as CTL and LTL widely used for expressing and verifying properties of transition systems.

We begin with states. Let σ^{f} be a multi-valued signature of constants called 'state variables', or more usually in AI terminology, *fluent constants*. Given a labelled transition system $\langle S, \mathbf{A}, R\rangle$ we add a valuation function which specifies, for every fluent constant $f \in \sigma^{\mathrm{f}}$ and every state $s \in S$, a value in $dom(f)$. Throughout this paper we shall be dealing with the *special case* of transition systems in which

- each state $s \in S$ is an interpretation of σ^{f}, $S \subseteq \mathrm{I}(\sigma^{\mathrm{f}})$.

The valuation function for fluent constants is then redundant: the state already specifies the value of each fluent constant in that state. It is often convenient to adopt the convention that an interpretation I of σ^{f} is represented by the set of atoms of σ^{f} that are satisfied by I. A state is then a (complete, and consistent) set of fluent atoms. We sometimes say a formula φ 'holds in' state s or 'is true in' state s as alternative ways of saying that s satisfies φ.

Although it is much less common, an idea employed in $\mathcal{C}+$ and in the Causal Calculator CCALC is that another category of constants and formulas — *action formulas* — can be interpreted on the transition labels/events of a transition system. So, let σ^{a} be a multi-valued signature of constants called *action constants*, disjoint from σ^{f}. Given a labelled transition system $\langle S, \mathbf{A}, R\rangle$ we add a valuation function for action constants which specifies, for every action constant $a \in \sigma^{\mathrm{f}}$ and every label/event $\varepsilon \in \mathbf{A}$, a value in $dom(a)$. Again, throughout the paper we shall be dealing with a special case, the case of labelled transition systems in which the set $\mathbf{A}$ of labels/events is the set of interpretations of

σ^{a}. In other words the transition systems of interest will be those of the form $\langle \sigma^{\mathrm{f}}, S, \mathrm{I}(\sigma^{\mathrm{a}}), R\rangle$, on which we will interpret various query languages of signature $\sigma^{\mathrm{f}} \cup \sigma^{\mathrm{a}}$, or variations thereof. $(\sigma^{\mathrm{f}}, \sigma^{\mathrm{a}})$ is the 'action signature' of the transition system.

Note that since a transition label/event ε is an interpretation of σ^{a}, it is meaningful to say that ε satisfies an action formula α ($\varepsilon \models_{\sigma^{\mathrm{a}}} \alpha$). When $\varepsilon \models_{\sigma^{\mathrm{a}}} \alpha$ we say that the event ε is of type α. When $\varepsilon \models_{\sigma^{\mathrm{a}}} \alpha$ we also say that the transition (s, ε, s') is a transition of type α.

Moreover, since a transition label is an interpretation of the action constants σ^{a}, it can also be represented by the set of atoms that it satisfies. The suggested reading of a transition label $\{a_1 = v_1, a_2 = v_2, \ldots, a_n = v_n\}$ for an action signature with action constants $a_1, a_2, \ldots, a_n$ is that it represents a composite action in which the elementary actions $a_1 = v_1, a_2 = v_2 \ldots, a_n = v_n$ are performed (or occur) concurrently. Where a is a Boolean action constant, $\neg a$, i.e. $a = \mathsf{f}$, can be read as indicating that action a is not performed; and where all action constants are Boolean, the action $\{a_1 = \mathsf{f}, \ldots, a_n = \mathsf{f}\}$ can be read as representing the 'null' event.

For example: suppose there are three agents, a, b, and c which can move in direction E, W, N, or S, or remain idle. Suppose (for the sake of an example) that they can also whistle as they move (they are trains, let us say). Let the action signature consist of action constants $\mathit{move}(a)$, $\mathit{move}(b)$, $\mathit{move}(c)$ with domains $\{E, W, N, S, \mathit{idle}\}$, and Boolean action constants $\mathit{whistle}(a)$, $\mathit{whistle}(b)$, $\mathit{whistle}(c)$. Then one possible interpretation of the action signature, and therefore one possible transition label, is

$$\{\mathit{move}(a) = E, \mathit{move}(b) = N, \mathit{move}(c) = \mathit{idle}, \mathit{whistle}(a), \neg \mathit{whistle}(b), \mathit{whistle}(c)\}$$

Because of the way that action formulas are evaluated on a transition (s, ε, s'), an action formula can also be regarded as expressing a property of the transition (s, ε, s') as a whole. For example, we might include in the action signature a Boolean action symbol *unusual* which is intended to be satisfied by transition labels/events which are regarded as unusual for some reason or another. In later sections we will employ the action constant trans (possible values: green and red) to indicate whether a given transition is 'permitted/acceptable' (green) or 'not permitted/unacceptable' (red).

Example (queries on states) For an example of one possible query language, consider the propositional language of signature σ^{f} extended with expressions of the form $[\alpha]\varphi$ where α is any formula of signature σ^{a}. $[\alpha]\varphi$ is intended to express that φ holds in every state following a transition of type α.

Let $\mathcal{T} = \langle \sigma^{\mathrm{f}}, S, \mathrm{I}(\sigma^{\mathrm{a}}), R\rangle$. Define the satisfaction relation $\models$ so that $\mathcal{T}, s \models f = v$ iff $s \models_{\sigma^{\mathrm{f}}} f = v$ for fluent atoms $f = v$, $\mathcal{T}, s \models \neg\varphi$ iff $\mathcal{T}, s \not\models \varphi$, $\mathcal{T}, s \models (\varphi \wedge \varphi')$ iff $\mathcal{T}, s \models \varphi$ and $\mathcal{T}, s \models \varphi'$, and so on as usual for the other truth functional connectives, and

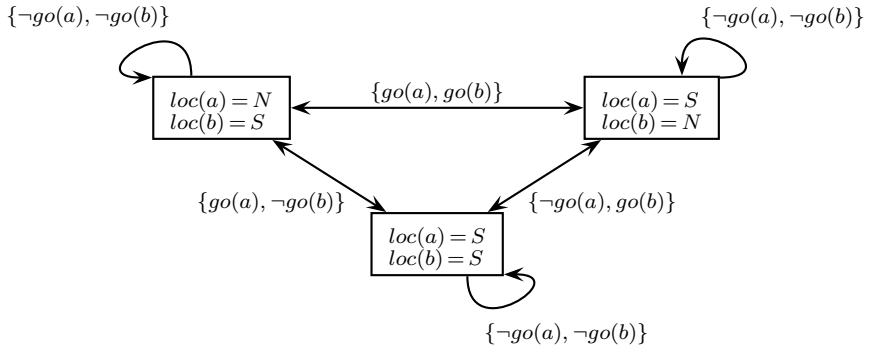
$$\mathcal{T}, s \models [\alpha]\varphi \text{ iff } \mathcal{T}, s' \models \varphi \text{ for every } s' \text{ such that } (s, \varepsilon, s') \in R \text{ and } \varepsilon \models_{\sigma^{\mathrm{a}}} \alpha.$$

The notation $[\alpha]\varphi$ is intended to be reminiscent of that of dynamic logic, except that in this language α is an action *formula* not a transition *label* as in dynamic

logic. In particular, 'action negation', which is problematic in dynamic logic, has a straightforward meaning here: $[\neg\alpha]\varphi$ simply says that φ holds in every state following a transition of type $\neg\alpha$.

We say (as usual) that a formula φ is *true in* the transition system $\mathcal{T} = \langle \sigma^{\mathrm{f}}, S, \mathrm{I}(\sigma^{\mathrm{a}}), R\rangle$, written $\mathcal{T} \models \varphi$, when $\mathcal{T}, s \models \varphi$ for every state s in $\mathcal{T}$.

Example Let σ^{f} be the set of fluent constants $\{loc(a), loc(b)\}$ with possible values $\{N, S\}$, and let σ^{a} be the set of Boolean action constants $\{go(a), go(b)\}$. Consider the transition system $\mathcal{T}$ depicted in the following diagram:



There is no state $\{loc(a) = N, loc(b) = N\}$ in $\mathcal{T}$ (for the sake of the example).

One can check by observation that, for example:

$$\begin{aligned}
\mathcal{T} &\models\ loc(a) = N \rightarrow [go(a)]\, loc(a) = S \\
\mathcal{T} &\models\ loc(a) = N \rightarrow [go(a)]\, [go(a)]\, loc(a) = N \\
\mathcal{T} &\models\ loc(a) = N \rightarrow [\neg go(a)]\, loc(a) = N \\
\mathcal{T} &\models\ loc(a) = N \rightarrow [go(b)]\, loc(a) = S \\
\mathcal{T} &\models\ (loc(a) = S\ \wedge\ loc(b) = S) \rightarrow [go(a) \wedge go(b)]\, \bot
\end{aligned}$$

Example (time-stamped queries) Query languages can also be interpreted on the paths ('runs') of a transition system. One candidate is the query language supported by the Causal Calculator CCalc. This uses propositional formulas of time-stamped fluent and action constants: the time-stamped fluent atom $f[i] = v$ represents that fluent atom $f = v$ holds at integer time i, or more precisely, that $f = v$ is satisfied by the state s_i of a path $s_0\, \varepsilon_0 \cdots \varepsilon_{i-1}\, s_i, \cdots$ of the transition system; the time-stamped atom $a[i] = v$ represents that action atom $a = v$ is satisfied by the transition label ε_i of a path $s_0\, \varepsilon_0 \cdots s_i\, \varepsilon_i\, s_{i+1}, \cdots$.

In general, given a multi-valued signature σ and a non-negative integer i, we write $\sigma[i]$ for the signature consisting of all constants of the form $c[i]$ where c is a constant of σ, with $dom(c[i]) = dom(c)$. For any non-negative integer m, we write σ_m for the signature $\sigma[0] \cup \cdots \cup \sigma[m]$. As usual, $Form(\sigma_m)$ is the set of formulas of signature σ_m.

The time-stamped query language used (as in CCalc) to express properties of paths of length m of a transition system with action signature $(\sigma^{\mathrm{f}}, \sigma^{\mathrm{a}})$ is

the propositional language of signature $\sigma^{\mathrm{f}}_m \cup \sigma^{\mathrm{a}}_{m-1}$. In other words, the query language is $\mathit{Form}(\sigma^{\mathrm{f}}_m \cup \sigma^{\mathrm{a}}_{m-1})$, whose formulas consist of:

- atoms $f[i]=v$ where $i \in 0\,..\,m$ and $f=v$ is a fluent atom of σ^{f};
- atoms $a[i]=v$ where $i \in 0\,..\,m{-}1$ and $a=v$ is an action atom of σ^{a};
- all truth-functional compounds of the above.

Let $\pi = s_0\,\varepsilon_0\,s_1\,\cdots\,s_{m-1}\,\varepsilon_{m-1}\,s_m$ be a path of length m of a transition system $\mathcal{T}$ of action signature $(\sigma^{\mathrm{f}}, \sigma^{\mathrm{a}})$. An atom $f[i]=v$ for any fluent constant f of σ^{f} and $0 \le i \le m$ is *true on* path π (or 'holds on' path π, or 'is satisfied by' path π), written $\mathcal{T}, \pi \models_m f[i]=v$, when $s_i \models_{\sigma^{\mathrm{f}}} f=v$; for action constants a of σ^{a} and $0 \le i < m$, $\mathcal{T}, \pi \models_m a[i]=v$ when $\varepsilon_i \models_{\sigma^{\mathrm{a}}} a=v$; and $\models_m$ is extended to formulas φ of signature $\sigma_m = \sigma^{\mathrm{f}}_m \cup \sigma^{\mathrm{a}}_{m-1}$ by the usual truth tables for the propositional connectives.

We will say φ is true on paths of length m of $\mathcal{T}$, written $\mathcal{T} \models_m \varphi$ when $\mathcal{T}, \pi \models_m \varphi$ for all paths of length m of $\mathcal{T}$.

The definition of the satisfaction relation $\models_m$ can be stated more concisely, as follows. First some notation (which will also be useful later).

Let $\pi[i]$ denote the ith component of a path π: that is, when $\pi = s_0\,\varepsilon_0\,s_1\,\cdots\,s_i\,\varepsilon_i\,s_{i+1}\,\ldots$, let $\pi[i] = s_i \cup \varepsilon_i$. Clearly, $\pi[i]$ is an interpretation of $\sigma^{\mathrm{f}} \cup \sigma^{\mathrm{a}}$ when π is a path of a transition system with action signature $(\sigma^{\mathrm{f}}, \sigma^{\mathrm{a}})$.

For any formula ψ of signature $\sigma^{\mathrm{f}} \cup \sigma^{\mathrm{a}}$, let $\psi[i]$ stand for the formula of signature $\sigma^{\mathrm{f}}[i] \cup \sigma^{\mathrm{a}}[i]$ obtained by time-stamping every constant in ψ with i, that is, replacing every constant c in ψ by the constant $c[i]$. Clearly, every formula φ of signature $\sigma_m = \sigma^{\mathrm{f}}_m \cup \sigma^{\mathrm{a}}_{m-1}$ is a truth-functional compound of formulas of the form $\psi[i]$ where $0 \le i \le m$ and ψ is a formula of signature $\sigma = \sigma^{\mathrm{f}} \cup \sigma^{\mathrm{a}}$.

Now, for any path π of length m of a transition system $\mathcal{T}$ of action signature $(\sigma^{\mathrm{f}}, \sigma^{\mathrm{a}})$, we have

$$\mathcal{T}, \pi \models_m \psi[i] \;\text{ iff }\; \pi[i] \models_\sigma \psi$$

Example (contd) Consider again the transition system $\mathcal{T}$ depicted in the figure above. We have, amongst other things:

$\mathcal{T} \models_1 \; (loc(a)[0]=N \;\wedge\; go(a)[0]) \rightarrow loc(a)[1]=S$

$\mathcal{T} \models_2 \; (loc(a)[0]=N \;\wedge\; go(a)[0] \;\wedge\; go(a)[1]) \rightarrow loc(a)[2]=N$

$\mathcal{T} \models_1 \; (loc(a)[0]=N \;\wedge\; \neg go(a)[0]) \rightarrow loc(a)[1]=N$

$\mathcal{T} \models_2 \; (loc(a)[0]=N \;\wedge\; go(b)[0] \;\wedge\; go(a)[1]) \rightarrow loc(a)[2]=N$

$\mathcal{T} \models_m \; (loc(a)[i]=N \;\wedge\; loc(a)[i{+}2]=N) \rightarrow (go(a)[i] \;\wedge\; go(a)[i{+}1]) \quad \text{for all } 0 \le i \le m{-}2$

$\mathcal{T} \models_m \; (loc(a)[i]=S \;\wedge\; loc(b)[i]=S) \rightarrow (\neg go(a)[i] \;\wedge\; \neg go(b)[i]) \quad \text{for all } 0 \le i \le m{-}1$

A wide variety of other languages can be interpreted on transition systems. In Section 9.6 we will show how the temporal logic CTL can be used to express properties of a transition system defined by an action description of $(\mathcal{C}+)^{++}$.

4 The Action Description Language $\mathcal{C}+$

The language $\mathcal{C}+$ has evolved through several versions. Here we follow the (definitive) presentation in (Giunchiglia et al., 2004) though with more emphasis on the transition system semantics and with some small notational and terminological differences. These differences are recorded in the text.

4.1 Syntax

An *action signature* $(\sigma^{\mathrm{f}}, \sigma^{\mathrm{a}})$ is a (non-empty) set σ^{f} of *fluent constants* and a (non-empty) set σ^{a} of *action constants*. Fluent constants are partitioned into *simple* fluent constants and *statically determined* fluent constants. *Simple fluent constants* are related to actions by *dynamic laws*, which specify how the values of simple fluents change in transitions from one state to another. The values of *statically determined fluents* can also vary from state to state but they are defined by *static laws* relating their values in a state to the values of other fluents in that state. Static laws can also express constraints on the values of simple fluent constants in a state.

Action constants are partitioned into *exogenous* action constants, which are used to name kinds of actions, and *transition attributes* which are used to represent attributes of actions and other properties of transitions. (This partitioning of action constants is not part of standard presentations of $\mathcal{C}+$. There, the declaration of exogenous actions is done by means of a special kind of $\mathcal{C}+$ law and not, as here, as part of the signature of the language. There is no practical difference between these two treatments, as will be explained later. We prefer to specify exogenous actions in the signature for uniformity with the treatment of fluent constants, but nothing of significance turns on this.)

A *fluent formula* is any truth-functional compound of fluent atoms (i.e., a formula of signature σ^{f}). An *action formula* is any formula of signature σ^{a} that contains at least one action constant. Notice that since $\top$ and $\bot$ are treated as 0-ary connectives, they are not action formulas, though $a \wedge \neg a$ and $a \vee \neg a$ are for any Boolean action constant a. The language also allows formulas of signature $\sigma^{\mathrm{f}} \cup \sigma^{\mathrm{a}}$.

The sentences of $\mathcal{C}+$ are of three kinds.

- A *static law* (or rule) is an expression of the form

$$\mathsf{caused}\ F\ \mathsf{if}\ G \tag{1}$$

  where F and G are fluent formulas (i.e., formulas of signature σ^{f}). F is the *head* of the rule and G is the *body*. Static laws are used to express constraints that hold in all states and to define statically determined fluents.
- A *fluent dynamic law* (or rule) is an expression of the form

$$\mathsf{caused}\ F\ \mathsf{if}\ G\ \mathsf{after}\ \psi \tag{2}$$

  where F and G are fluent formulas, and ψ is a formula of signature $\sigma^{\mathrm{f}} \cup \sigma^{\mathrm{a}}$, with the restriction that statically determined fluent constants may not appear in the head of the rule, F. We will say $G\ \mathsf{after}\ \psi$ is the body of the rule.

Informally, in a transition (s, ε, s'), formulas F and G are evaluated at s' (the resulting state), fluent atoms in ψ are evaluated at s (i.e., in the state immediately before the transition), and action atoms in ψ are evaluated on the transition label ε itself. It may be helpful to note that any set of fluent dynamic laws can be written equivalently as a set of fluent dynamic laws of the form

$$\mathsf{caused}\ F\ \mathsf{if}\ G\ \mathsf{after}\ H \wedge \alpha$$

where H is a fluent formula and α is an action formula, and this is the form that will appear most frequently in subsequent sections. Fluent dynamic laws are primarily used to express how the values of fluents are affected by different kinds of actions, and to specify which of the fluents are 'inertial', that is, which fluents are such that their values persist by default from one state to the next. The expression

$$\alpha\ \mathsf{causes}\ F\ \mathsf{if}\ H$$

is shorthand for a fluent dynamic law of the form: $\mathsf{caused}\ F\ \mathsf{if}\ \top\ \mathsf{after}\ H \wedge \alpha$.

- An *action dynamic law* (or rule) is an expression of the form

$$\mathsf{caused}\ \alpha\ \mathsf{if}\ \psi \tag{3}$$

where α is an action formula (i.e., a formula of signature σ^{a}) and ψ is any formula of signature $\sigma^{\mathrm{f}} \cup \sigma^{\mathrm{a}}$. α is the head of the rule and ψ is the body.

Action dynamic laws are used to express, among other things, that any transition of type α must also be of type α' ($\mathsf{caused}\ \alpha'\ \mathsf{if}\ \alpha$), or that any transition from a state satisfying fluent formula G must be of type β ($\mathsf{caused}\ \beta\ \mathsf{if}\ G$). Examples will be provided in later sections.

An *action description* is a set of static and dynamic laws.

In the remainder of this presentation we will omit the keyword caused when writing causal laws. This is simply to shorten expressions and avoid linebreaks.

Example The effects of toggling a switch between on and off can be represented by a simple Boolean fluent constant *on* and an (exogenous) Boolean action constant *toggle* and the following pair of laws:

$$\mathit{toggle}\ \mathsf{causes}\ \mathit{on}\ \mathsf{if}\ \neg \mathit{on}$$
$$\mathit{toggle}\ \mathsf{causes}\ \neg \mathit{on}\ \mathsf{if}\ \mathit{on}$$

which, recall, are shortand for the following fluent dynamic laws:

$$\mathit{on}\ \mathsf{if}\ \top\ \mathsf{after}\ \mathit{toggle} \wedge \neg \mathit{on}$$
$$\neg \mathit{on}\ \mathsf{if}\ \top\ \mathsf{after}\ \mathit{toggle} \wedge \mathit{on}$$

Note that default persistence ('inertia') of fluent values is not a built-in feature of the $\mathcal{C}+$ language. One specifies explicitly which fluents are 'inertial' by means of a $\mathcal{C}+$ law of the form

$$\mathsf{inertial}\ \ f$$

This is shorthand for the set of fluent dynamic laws of the form

$$f = v\ \mathsf{if}\ f = v\ \mathsf{after}\ f = v, \quad \text{for every } v \in \mathit{dom}(f)$$

How this law works to express default persistence of $f = v$ will become clearer when we look at the semantics of $\mathcal{C}+$ laws. Notice that a statically determined fluent constant cannot be inertial, since inertial declarations are fluent dynamic laws, and no statically determined fluent can be in the head of a fluent dynamic law.

Early versions of the language also included a separate category of *rigid constants* in the action signature, used to represent fluents whose values are constant and do not vary from state to state. In the latest presentations (Giunchiglia et al., 2004), rigid constants are not features of an action signature but of an action description. A fluent constant c is rigid (relative to an action description D) if the value of c is the same in every state of the transition system defined by D. Trivial fluent constants are clearly rigid, but there may also be rigid constants which are not trivial.

Definite action descriptions

Of particular interest are *definite* action descriptions.

An action description D is *definite* when, for all static laws, fluent dynamic laws, and action dynamic laws in D:

- the head of every law is either a fluent atom (resp., action atom) or the symbol $\bot$, and
- no atom is the head of infinitely many laws of D.

(Note that, as defined above, a Boolean fluent constant p and its negation $\neg p$, and a Boolean action constant a and its negation $\neg a$ are atoms, and hence are included in the definition.)

4.2 Semantics

Consider an action description D of $\mathcal{C}+$ of signature $(\sigma^{\mathrm{f}}, \sigma^{\mathrm{a}})$.

- A *state* is an interpretation of the fluent constants σ^{f} that is closed under the static laws of D, that is to say, that satisfies $G \rightarrow F$ for every static law F if G in D. A state must further satisfy certain restrictions concerning statically determined fluents which we will identify presently.
- A *transition label* or *event* is an interpretation of the action constants σ^{a}. A transition label must satisfy certain constraints concerning the exogenous action constants, as identified below.
- A *transition* is a triple (s, ε, s') where s and s' are states and ε is a transition label/event. A transition *defined by* an action description D must also satisfy the fluent dynamic laws and the action dynamic laws of D, in a sense to be defined below.

The *transition system* defined by an action description D is the transition system $\langle \sigma^{\mathrm{f}}, S, \mathrm{I}(\sigma^{\mathrm{a}}), R \rangle$ which has the states of D as its states S, and which has $(s, \varepsilon, s') \in R$ when (s, ε, s') is a transition of D.

Semantics: Definitions

Let $T_{static}(s)$ stand for the heads of all static laws in D whose bodies are satisfied by s; let $E(s, \varepsilon, s')$ stand for the heads of all fluent dynamic laws in D whose bodies are satisfied by the transition (s, ε, s'); and let $A(\varepsilon, s)$ stand for the heads of all action dynamic laws whose bodies are satisfied by the transition (s, ε, s'). More precisely:

$$\begin{array}{rcl} T_{static}(s) & =_{\text{def}} & \{F \mid F \text{ if } G \text{ is in } D,\ s \models G\} \\ E(s, \varepsilon, s') & =_{\text{def}} & \{F \mid F \text{ if } G \text{ after } \psi \text{ is in } D,\ s' \models G,\ s \cup \varepsilon \models \psi\} \\ A(\varepsilon, s) & =_{\text{def}} & \{\alpha \mid \alpha \text{ if } \psi \text{ is in } D,\ s \cup \varepsilon \models \psi\} \end{array}$$

(Throughout this section we omit subscripts on $\models$ whenever context allows. Notice that since σ^{f} and σ^{a} are disjoint, $s \models_{\sigma^{\text{f}}} F$ iff $s \models_{\sigma^{\text{f}} \cup \sigma^{\text{a}}} F$ for any fluent formula F, and $\varepsilon \models_{\sigma^{\text{a}}} \alpha$ iff $\varepsilon \models_{\sigma^{\text{f}} \cup \sigma^{\text{a}}} \alpha$ for any action formula α.)

In the absence of statically determined fluents, an interpretation s of σ^{f} is a state of D iff it is closed under all the static laws, i.e., iff $s \models T_{static}(s)$. In the presence of statically determined fluents, the characterisation of what constitutes a state is a little more complicated.

An interpretation s of σ^{f} is a *state* of D when

- $s \models T_{static}(s)$
- there is no other interpretation s' of σ^{f} which agrees with s on the interpretation of all simple fluents and is such that $s' \models T_{static}(s)$.

Alternatively: let $Simple(s)$ denote the simple fluent atoms that are satisfied by an interpretation s of σ^{f}. Clearly $s \models Simple(s)$ for any s. So then we can say the following.

Definition 2 *Let D be an action description with fluent signature σ^{f}. An interpretation s of σ^{f} is a* state *of D when*

$$\{s' \in \mathrm{I}(\sigma^{\text{f}}) \mid s' \models T_{static}(s) \cup Simple(s)\} = \{s\}$$

In words: when $s \models T_{static}(s) \cup Simple(s)$ and there is no other interpretation s' of σ^{f} such that $s' \models T_{static}(s) \cup Simple(s)$.

The rationale behind these definitions, especially that of the uniqueness condition, is perhaps far from obvious. It is a direct application of the methods of 'nonmonotonic causal theories' (see Section 5 below) in which the language $\mathcal{C}+$ has its origins. For convenience we will write $X \mathrel{!\models} \varphi$ to denote that X is the only interpretation that satisfies a formula (or set of formulas) φ. In this notation, an interpretation s of σ^{f} is a state of D when

$$s \mathrel{!\models} T_{static}(s) \cup Simple(s)$$

A transition (s, ε, s') satisfies the action dynamic laws of D when:

- $\varepsilon \models A(\varepsilon, s)$
- there is no other interpretation ε' of σ^{a} which agrees with ε on the interpretation of all exogenous action constants and is such that $\varepsilon' \models A(\varepsilon, s)$.

Again, let $Exog(\varepsilon)$ denote the exogenous atoms of σ^{a} that are satisfied by an interpretation ε of σ^{a}. Clearly, $\varepsilon \models Exog(\varepsilon)$. The transition (s, ε, s') satisfies the action dynamic laws of D when

$$\varepsilon \mathrel{!\models} A(\varepsilon, s) \cup Exog(\varepsilon)$$

that is, when $\{\varepsilon' \in \mathrm{I}(\sigma^{\mathrm{a}}) \mid \varepsilon' \models A(\varepsilon, s) \cup Exog(\varepsilon)\} = \{\varepsilon\}$.

(s, ε, s') is a *transition defined by* an action description D (or simply: is a transition of D) if s is a state of D, the transition (s, ε, s') satisfies the action dynamic laws of D, and the resulting state s' is the only interpretation of σ^{f} that satisfies all formulas $T_{static}(s') \cup E(s, \varepsilon, s')$. (Early presentations of $\mathcal{C}+$, e.g. (Giunchiglia and Lifschitz, 1998; Giunchiglia et al., 2001) referred to this as a *causally explained* transition.)

Definition 3 *Let D be an action description of $\mathcal{C}+$ of signature $(\sigma^{\mathrm{f}}, \sigma^{\mathrm{a}})$. Let s and s' be interpretations of σ^{f}, and ε an interpretation of σ^{a}. (s, ε, s') is a transition of D iff*

- $s \mathrel{!\models} T_{static}(s) \cup Simple(s)$
- $\varepsilon \mathrel{!\models} A(\varepsilon, s) \cup Exog(\varepsilon)$
- $s' \mathrel{!\models} T_{static}(s') \cup E(s, \varepsilon, s')$

In words: (s, ε, s') is a transition of D iff s is a state of D, ε is the only model of $A(\varepsilon, s) \cup Exog(\varepsilon)$, and s' is the only model of $E(s, \varepsilon, s')$ that also satisfies the static laws of D.

The uniqueness condition in the above definition does *not* imply that all events are deterministic, as will be illustrated in examples later.

The *transition system* defined by an action description D is the transition system $\langle \sigma^{\mathrm{f}}, S, \mathrm{I}(\sigma^{\mathrm{a}}), R \rangle$ which has the states of D as its states S, and which has $(s, \varepsilon, s') \in R$ when (s, ε, s') is a transition of D. We will say that π is a path of D when π is a path of the transition system defined by D.

Notice that the last condition of Definition 3 makes no mention of $Simple(s')$. It is not obvious therefore that the definition is well formed, that is, that the resulting state s' of every transition (s, ε, s') defined by D is guaranteed to be a state of D. It is easy to check that this is so. It relies on the fact that formulas in $E(s, \varepsilon, s')$ contain only simple and no statically determined fluent constants. (It might appear there is a simple argument relying on the observation that $E(s, \varepsilon, s') \subseteq Simple(s')$. But that is not so, because in the general case $E(s, \varepsilon, s')$ is a set of *formulas* not a set of *atoms*.)

Proposition 4 *If (s, ε, s') is a transition defined by an action description D then s' is a state of D.*

Proof. We show that if s' is the only model of $T_{static}(s') \cup E(s, \varepsilon, s')$ then s' is the only model of $T_{static}(s') \cup Simple(s')$.

Clearly s' is a model of $T_{static}(s') \cup Simple(s')$ because $s' \models T_{static}(s')$ (from the assumption) and $s' \models Simple(s')$ (by definition). So it remains to show that s' is the only such model.

Notice first that a state s' can be represented in the form $s'_{\rm s} \cup \delta'$ where $s'_{\rm s}$ is an interpretation of the simple fluent constants and δ' is an interpretation of the statically determined fluent constants. We then have $\mathit{Simple}(s'_{\rm s} \cup \delta') = s'_{\rm s}$. So suppose that $s'_{\rm s} \cup \delta'$ is the only model of $T_{\mathit{static}}(s'_{\rm s} \cup \delta') \cup E(s, \varepsilon, s'_{\rm s} \cup \delta')$. We need to show that $s'_{\rm s} \cup \delta'$ is the only model of $T_{\mathit{static}}(s'_{\rm s} \cup \delta') \cup \mathit{Simple}(s'_{\rm s} \cup \delta')$, that is, that there is no other interpretation $s''_{\rm s} \cup \delta''$ such that $s''_{\rm s} \cup \delta'' \models T_{\mathit{static}}(s'_{\rm s} \cup \delta') \cup \mathit{Simple}(s'_{\rm s} \cup \delta')$.

For any $s''_{\rm s} \cup \delta''$, $s''_{\rm s} \cup \delta'' \models \mathit{Simple}(s'_{\rm s} \cup \delta')$ if and only if $s''_{\rm s} = s'_{\rm s}$, because $\mathit{Simple}(s'_{\rm s} \cup \delta') = s'_{\rm s}$.

So it remains to show that there is no other interpretation $\delta'' \neq \delta'$ of the statically determined fluent constants such that $s'_{\rm s} \cup \delta'' \models T_{\mathit{static}}(s'_{\rm s} \cup \delta')$.

Suppose there were such a δ''. $s'_{\rm s} \cup \delta' \models E(s, \varepsilon, s'_{\rm s} \cup \delta')$, and $E(s, \varepsilon, s'_{\rm s} \cup \delta')$ contains no statically determined fluents, so $s'_{\rm s} \cup \delta'' \models E(s, \varepsilon, s'_{\rm s} \cup \delta')$ also. Now we have $s'_{\rm s} \cup \delta'' \models T_{\mathit{static}}(s'_{\rm s} \cup \delta') \cup E(s, \varepsilon, s'_{\rm s} \cup \delta')$, which contradicts the assumption that $s'_{\rm s} \cup \delta'$ is the only such interpretation. □

In future sections, (in particular in Section 6), we will be primarily concerned with paths/runs of the transition system defined by D. Proposition 4 can also be applied to simplify the the characterisation of paths of D, as follows. $(s_{i-1}, \varepsilon_{i-1}, s_i)$ and $(s_i, \varepsilon_i, s_{i+1})$ are both transitions of D when

$$\begin{aligned} s_{i-1} &\not\models T_{\mathit{static}}(s_{i-1}) \cup \mathit{Simple}(s_{i-1}), \\ \varepsilon_{i-1} &\not\models A(\varepsilon_{i-1}, s_{i-1}) \cup \mathit{Exog}(\varepsilon_{i-1}), \\ s_i &\not\models T_{\mathit{static}}(s_i) \cup E(s_{i-1}, \varepsilon_{i-1}, s_i), \end{aligned}$$

and

$$\begin{aligned} s_i &\not\models T_{\mathit{static}}(s_0) \cup \mathit{Simple}(s_i), \\ \varepsilon_i &\not\models A(\varepsilon_i, s_i) \cup \mathit{Exog}(\varepsilon_i), \\ s_{i+1} &\not\models T_{\mathit{static}}(s_{i+1}) \cup E(s_i, \varepsilon_i, s_{i+1}). \end{aligned}$$

But by Proposition 4, $s_i \not\models T_{\mathit{static}}(s_i) \cup E(s_{i-1}, \varepsilon_{i-1}, s_i)$ already implies $s_i \not\models T_{\mathit{static}}(s_0) \cup \mathit{Simple}(s_i)$ (s_i is a state of D), and so this condition can be dropped. We thus have the following.

Proposition 5 *Let D be a $\mathcal{C}+$ action description of signature $(\sigma^{\rm f}, \sigma^{\rm a})$.*

$s_0\, \varepsilon_0\, s_1 \cdots \varepsilon_{m-1}\, s_m$ is a path of length $m \geq 0$ of the transition system defined by D iff $s_0, s_1, \ldots, s_m$ are interpretations of $\sigma^{\rm f}$ and $\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_{m-1}$ are interpretations of $\sigma^{\rm a}$ such that

$$\begin{aligned} s_0 &\not\models T_{\mathit{static}}(s_0) \cup \mathit{Simple}(s_0) \\ \varepsilon_0 &\not\models A(\varepsilon_0, s_0) \cup \mathit{Exog}(\varepsilon_0) \\ s_1 &\not\models T_{\mathit{static}}(s_1) \cup E(s_0, \varepsilon_0, s_1) \\ &\vdots \end{aligned}$$

$$
\begin{aligned}
\varepsilon_i \;&!\models A(\varepsilon_i, s_i) \cup \mathit{Exog}(\varepsilon_i) \\
s_{i+1} \;&!\models T_{\mathit{static}}(s_{i+1}) \cup E(s_i, \varepsilon_i, s_{i+1}) \\
&\vdots \\
\varepsilon_{m-1} \;&!\models A(\varepsilon_{m-1}, s_{m-1}) \cup \mathit{Exog}(\varepsilon_{m-1}) \\
s_m \;&!\models T_{\mathit{static}}(s_m) \cup E(s_{m-1}, \varepsilon_{m-1}, s_m)
\end{aligned}
$$

Proof. In the preceding discussion. The cases $m = 0$ and $m = 1$ are trivial. □

Semantics: Definite action descriptions

When D is a definite action description, the conditions under which an interpretation s of σ^{f} is a state, and the conditions under which (s, ε, s') is a transition defined by D, can be simplified very considerably. For then $T_{\mathit{static}}(s)$, $E(s, \varepsilon, s')$, and $A(\varepsilon, s)$ are all sets of *atoms*, possibly also containing the element $\bot$. And if we represent an interpretation s of σ^{f} by the set of atoms of σ^{f} that are satisfied by s, the condition that s is the *only* interpretation of σ^{f} that satisfies all formulas $T_{\mathit{static}}(s) \cup \mathit{Simple}(s)$ is equivalently stated as $s = T_{\mathit{static}}(s) \cup \mathit{Simple}(s) \cup \mathit{Trivial}(\sigma^{\mathrm{f}})$, where $\mathit{Trivial}(\sigma^{\mathrm{f}})$ denotes the set (possibly empty) of the trivial atoms of σ^{f}. Trivial atoms complicate the account slightly but since they can be used to represent *rigid* fluent constants we will keep open the possibility that they are present.

Informally: suppose first that $s = T_{\mathit{static}}(s) \cup \mathit{Simple}(s)$. Any interpretation s' of σ^{f} satisfying $T_{\mathit{static}}(s) \cup \mathit{Simple}(s)$ must also be such that $T_{\mathit{static}}(s) \cup \mathit{Simple}(s) \subseteq s'$; every interpretation s' of σ^{f} must also satisfy $\mathit{Trivial}(\sigma^{\mathrm{f}})$. However, s' cannot contain any additional atoms of σ^{f} not in $T_{\mathit{static}}(s) \cup \mathit{Simple}(s) \cup \mathit{Trivial}(\sigma^{\mathrm{f}})$ because s is already a complete set of atoms and s' would not then be an interpretation of σ^{f}. So s is unique. We can generalise the above informal argument, as follows.

Proposition 6 *Let X be a set of atoms of signature σ. Let I be an interpretation of σ, represented as a set of atoms. Let $\mathit{Trivial}(\sigma)$ denote the set (possibly empty) of trivial atoms of σ. Then I is the unique interpretation of σ satisfying X iff $I = X \cup \mathit{Trivial}(\sigma)$.*

Proof. Left-to-right: Suppose the set of atoms I is the only interpretation of σ such that $I \models X$. Then it follows that $X \models_\sigma I$ (i.e., atoms I are all satisfied in all models of X): there is only one interpretation of σ that satisfies X, that interpretation is I, and clearly $I \models I$. $I \models X$ so X is σ-satisfiable, and by Proposition 1, $X \models_\sigma I$ implies $I \subseteq X \cup \mathit{Trivial}(\sigma)$. Further, since $I \models X$ we have $X \subseteq I$. And clearly $\mathit{Trivial}(\sigma) \subseteq I$. So we have $X \cup \mathit{Trivial}(\sigma) \subseteq I \subseteq X \cup \mathit{Trivial}(\sigma)$.
Right-to-left: Suppose $I = X \cup \mathit{Trivial}(\sigma)$. Clearly I satisfies X because $X \subseteq I$. To show I is unique, suppose $I' \models X$ for some other interpretation I' of σ, also represented as a set of atoms. We have $X \subseteq I'$, and so $I \subseteq X \subseteq I'$, and hence $I \subseteq I'$. But for interpretations I and I', $I \subseteq I'$ iff $I' = I$. □

Applying Proposition 6 to the definition of a state of D, we obtain the following concise definition for the case where D is a definite action description.

Proposition 7 *Let D be a definite action description of signature $(\sigma^{\mathrm{f}}, \sigma^{\mathrm{a}})$. Let s be a set of atoms of σ^{f} representing an interpretation of σ^{f}. s is a state of D iff*

$$s = T_{static}(s) \cup Simple(s) \cup Trivial(\sigma^{\mathrm{f}})$$

□

Notice that the above characterisation of a state does not hold for an *arbitrary* set s of atoms of σ^{f} — s must be an interpretation (a consistent and complete set of atoms) of σ^{f}.

It is perhaps helpful to see that the conditions for an interpretation s of σ^{f} to be a state of D can be equivalently stated as follows. First, in the case where there are no trivial atoms in σ^{f}:

- $T_{static}(s) \subseteq s$
- $s - Simple(s) = T_{static}(s) - Simple(s)$

In other words, the state s must satisfy the static laws of D, and the set of statically determined atoms in s, $s - Simple(s)$, must be completely determined by the static laws of D. When $Trivial(\sigma^{\mathrm{f}}) \neq \emptyset$:

- $T_{static}(s) \subseteq s$
- $s - Simple(s) = \big(T_{static}(s) \cup Trivial(\sigma^{\mathrm{f}})\big) - Simple(s)$

By similar applications of Proposition 6, for any definite action description D, the condition that a set s' of atoms represents the only interpretation of σ^{f} satisfying $T_{static}(s') \cup E(s, \varepsilon, s')$ is equivalently stated as $s' = T_{static}(s') \cup E(s, \varepsilon, s') \cup Trivial(\sigma^{\mathrm{f}})$; and the condition that a set ε of action atoms represents the only interpretation of σ^{a} satisfying $A(\varepsilon, s) \cup Exog(\varepsilon)$ is equivalently stated as $\varepsilon = A(\varepsilon, s) \cup Exog(\varepsilon) \cup Trivial(\sigma^{\mathrm{a}})$. So then:

Proposition 8 *Let D be a definite action description of signature $(\sigma^{\mathrm{f}}, \sigma^{\mathrm{a}})$. (s, ε, s') is a transition of D iff s and s' are interpretations of σ^{f} and ε is an interpretation of σ^{a} such that:*

- $s = T_{static}(s) \cup Simple(s) \cup Trivial(\sigma^{\mathrm{f}})$ *(s is a state of D)*
- $\varepsilon = A(\varepsilon, s) \cup Exog(\varepsilon) \cup Trivial(\sigma^{\mathrm{a}})$
- $s' = T_{static}(s') \cup E(s, \varepsilon, s') \cup Trivial(\sigma^{\mathrm{f}})$

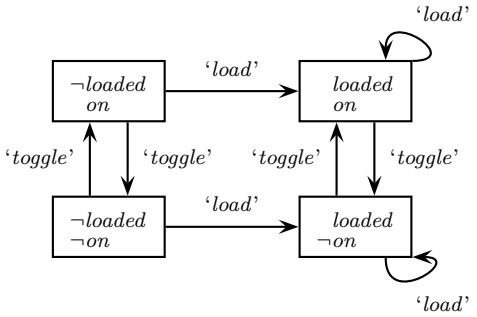
4.3 Example

Signature: simple Boolean fluent constants *loaded*, *on*; exogenous Boolean action constants *load*, *toggle*.

There are no static laws in this first example.

inertial *loaded*
inertial *on*

load causes *loaded*
toggle causes *on* if $\neg on$
toggle causes $\neg on$ if *on*



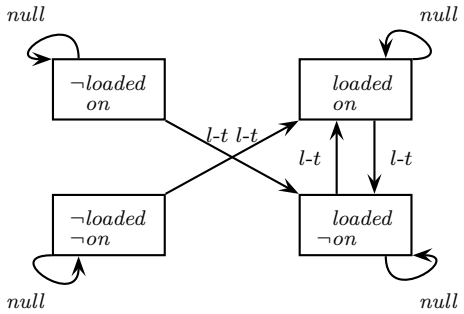
(Action *load* is supposed to mean something like 'ensure that loaded'. Otherwise we would change the action description to *load* causes *loaded* if $\neg loaded$.)

In the diagram, transition labels '*load*' and '*toggle*' are shorthand for $\{load, \neg toggle\}$ and $\{\neg load, toggle\}$, respectively.

Let's consider state $\{\neg loaded, on\}$ and event $\{load, \neg toggle\}$. There are no static laws so we can ignore T_{static} in this example, and there are no action dynamic laws to consider.

- Is $(\{\neg loaded, on\}, \{load, \neg toggle\}, \{loaded, on\})$ a transition defined by this action description?
  Yes, because $E(\{\neg loaded, on\}, \{load, \neg toggle\}, \{loaded, on\}) = \{loaded, on\}$.
- Is $(\{\neg loaded, on\}, \{load, \neg toggle\}, \{loaded, \neg on\})$ a transition defined by this action description??
  No, because $E(\{\neg loaded, on\}, \{load, \neg toggle\}, \{loaded, \neg on\}) = \{loaded\} \neq \{loaded, \neg on\}$.

There are two other kinds of transitions, not shown in the diagram above. They are transitions with labels $\{load, toggle\}$ and $\{\neg load, \neg toggle\}$.



The label *l-t* in the diagram is shorthand for $\{load, toggle\}$ and *null* is shorthand for $\{\neg load, \neg toggle\}$.

If we wanted to eliminate the *null* events, we could add the following law to the action description:

$$\bot \text{ if } \top \text{ after } \neg load \wedge \neg toggle$$

for which there is a standard abbreviation in $\mathcal{C}+$:

$$\text{nonexecutable } \neg load \wedge \neg toggle$$

Remark: the reading of causes The choice of the abbreviation causes in $\mathcal{C}+$ can be misleading. For example, we might observe that whenever Jack leaves work, he is tired. We can express this by means of a fluent dynamic law

$$tired(Jack) \text{ if } \top \text{ after } leaves\_work(Jack)$$

In abbreviated form this is

$$leaves\_work(Jack) \text{ causes } tired(Jack)$$

But we would surely not want to say that Jack's action of leaving work *causes* him to be tired.

We will retain the abbreviation causes in what follows since most other accounts and examples of $\mathcal{C}+$ make extensive use of it. However, we emphasise that the reading of causes in examples to follow should not be given any strong of reading of causation: it is the unabbreviated fluent dynamic law that is intended.

4.4 Abbreviations

The language $\mathcal{C}+$ provides various abbreviations, of which causes, inertial, and nonexecutable are the most common. Here is the full list, as presented in (Giunchiglia et al., 2004, Appendix B).

In the following, F and G are fluent formulas, α and α' are action formulas, ψ is a formula, f is a fluent constant, and a is an action constant.

F	F if $\top$
α	α if $\top$
F after ψ	F if $\top$ after ψ

default F	F if F
default F if G	F if $F \wedge G$
inertial f	$f = v$ if $f = v$ after $f = v$, for all $v \in dom(f)$
inertial f if ψ	$f = v$ if $f = v$ after $f = v \wedge \psi$, for all $v \in dom(f)$
α causes F	F if $\top$ after α
α causes F if ψ	F if $\top$ after $\alpha \wedge \psi$
nonexecutable α	$\bot$ if $\top$ after α (or: α causes $\bot$)
nonexecutable α if ψ	$\bot$ if $\top$ after $\alpha \wedge \psi$ (or: α causes $\bot$ if ψ)
α may cause F	F if F after α
α may cause F if ψ	F if F after $\alpha \wedge \psi$

The last of these, may cause, is used for specifying the effects of non-deterministic actions. An example will follow later.

The form nonexecutable α is, strictly, not allowed according to (Giunchiglia et al., 2004) but is used in that paper.

We also have, where a is an action constant and f a fluent constant:

exogenous a	$a = v$ if $a = v$, for all $v \in dom(a)$ (or: default $a = v$, for all $v \in dom(a)$)
exogenous a if ψ	$a = v$ if $a = v \wedge \psi$, for all $v \in dom(a)$ (or: default $a = v$ if ψ, for all $v \in dom(a)$)
exogenous f	$f = v$ if $f = v$, for all $v \in dom(f)$ (or: default $f = v$, for all $v \in dom(f)$)
exogenous f if G	$f = v$ if $f = v \wedge G$, for all $v \in dom(f)$ (or: default $f = v$ if G, for all $v \in dom(f)$)

Exogenous fluent constants are those whose values can vary from one state to another, but not under the effects of actions in the action description. Whether it is raining could be represented by an exogenous (Boolean) fluent constant, for example.

For reference, here are the remaining abbreviations given in (Giunchiglia et al., 2004, Appendix B). They are not used in this paper.

α causes α' if ψ	α' if $\alpha \wedge \psi$
α may cause α' if ψ	α' if $\alpha' \wedge \alpha \wedge \psi$
default α	α if α
default α if ψ	α if $\alpha \wedge \psi$
default F if G after ψ	F if $F \wedge G$ after ψ
default F after ψ	default F if $\top$ after ψ (or: F if F after ψ)
constraint F	$\bot$ if $\neg F$
constraint F after ψ	$\bot$ if $\neg F$ after ψ
rigid f	$\bot$ if $f \neq v$ after $f = v$, for all $v \in dom(f)$ (or: constraint $f = v$ after $f = v$, for all $v \in dom(f)$)
always ψ	$\bot$ after $\neg\psi$

For f a *statically determined* Boolean fluent constant, and a a Boolean action constant:

F if G unless f	F if $G \wedge \neg f$ default $\neg f$
F if G after ψ unless a	F if G after $\psi \wedge \neg a$ default $\neg a$
α if ψ unless a	α if $\psi \wedge \neg a$ default $\neg a$

Note that many of the above abbreviations are ambiguous as given in (Giunchiglia et al., 2004, Appendix B), in the special case that components are $\top$ or $\bot$.

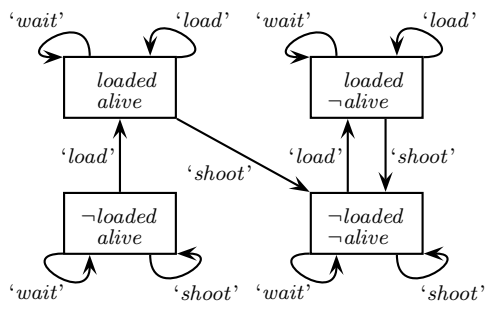
4.5 **Example** ('Yale Shooting Problem')

Signature: simple Boolean fluent constants *loaded*, *alive*; exogenous Boolean action constants *load*, *shoot*, *wait*.

inertial *loaded*
inertial *alive*

load causes *loaded*
shoot causes $\neg$*alive* if *loaded*
shoot causes $\neg$*loaded*

nonexecutable *shoot* $\wedge$ *load*
nonexecutable *wait* $\wedge$ *shoot*
nonexecutable *wait* $\wedge$ *load*
nonexecutable $\neg$*wait* $\wedge$ $\neg$*shoot* $\wedge$ $\neg$*load*



The diagram uses the same shorthand convention for transition labels as used earlier.

It is not possible to load and shoot a gun at the same time: *shoot* $\wedge$ *load* events are eliminated by the first of the nonexecutable laws.

Alternatively, we could dispense with the action constant *wait* and represent it instead by the 'null' event $\{\neg shoot, \neg load\}$. The last three lines of the action description could then be deleted.

Consider now the query languages introduced in Section 3.3. Let D_{YSP} be the action description above. We write $D_{\mathrm{YSP}} \models \varphi$ (resp. $D_{\mathrm{YSP}} \models_m \psi$) when $\mathcal{T}_{\mathrm{YSP}}$ is the transition system defined by D_{YSP} and $\mathcal{T}_{\mathrm{YSP}} \models \varphi$ (resp. $\mathcal{T}_{\mathrm{YSP}} \models_m \psi$). One can see from the diagram above that we have, for example:

$$D_{\mathrm{YSP}} \models [shoot]\,[load]\,\neg alive$$
$$D_{\mathrm{YSP}} \models_3 (load[0] \wedge shoot[2]) \rightarrow \neg alive[3]$$

and indeed

$$D_{\mathrm{YSP}} \models_m (load[i] \wedge shoot[j]) \rightarrow \neg alive[k] \quad \text{for any } 0 \leq i < j < k \leq m$$

4.6 Example

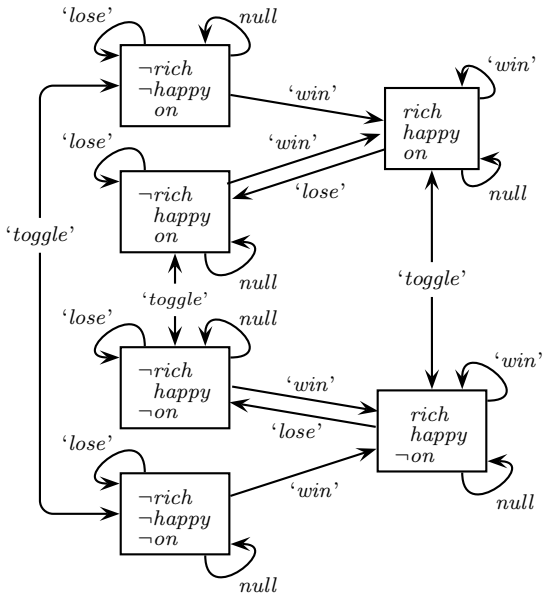
This is a completely artificial example, just for the sake of illustrating how static laws work as constraints on simple fluents.

Signature: simple Boolean fluent constants *rich*, *happy*, *on*; exogenous Boolean action constants *win*, *lose*, *toggle*.

inertial *rich*
inertial *on*
inertial *happy*
win causes *rich*
lose causes $\neg$*rich*
toggle causes *on* if $\neg$*on*
toggle causes $\neg$*on* if *on*

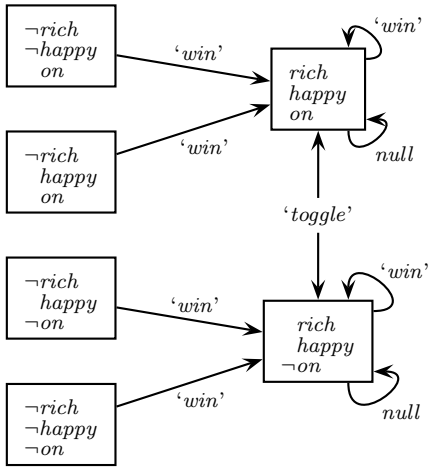
happy if *rich* (a static law)

nonexecutable *lose* $\wedge$ *win*



Because of the static law, there are only 6 states not $2^3 = 8$. The diagram does not show the transitions with labels $\{toggle, win, \neg lose\}$ and $\{toggle, \neg win, lose\}$. This is just to reduce clutter.

What if we drop the law that specifies *happy* is inertial? In that case, since there is nothing to 'cause' $\neg happy$ ($\neg happy \notin T_{static}(s)$ for any s and $\neg happy \notin E(s, \varepsilon, s')$ for any transition (s, ε, s')), there can be no transitions to any state where $\neg happy$ holds, and by a similar argument, no transitions to *happy* states where *rich* does not hold. There are thus no '*lose*' transitions, and some *null* and '*toggle*' transitions are eliminated also, as shown in the following diagram.



without inertial *happy*

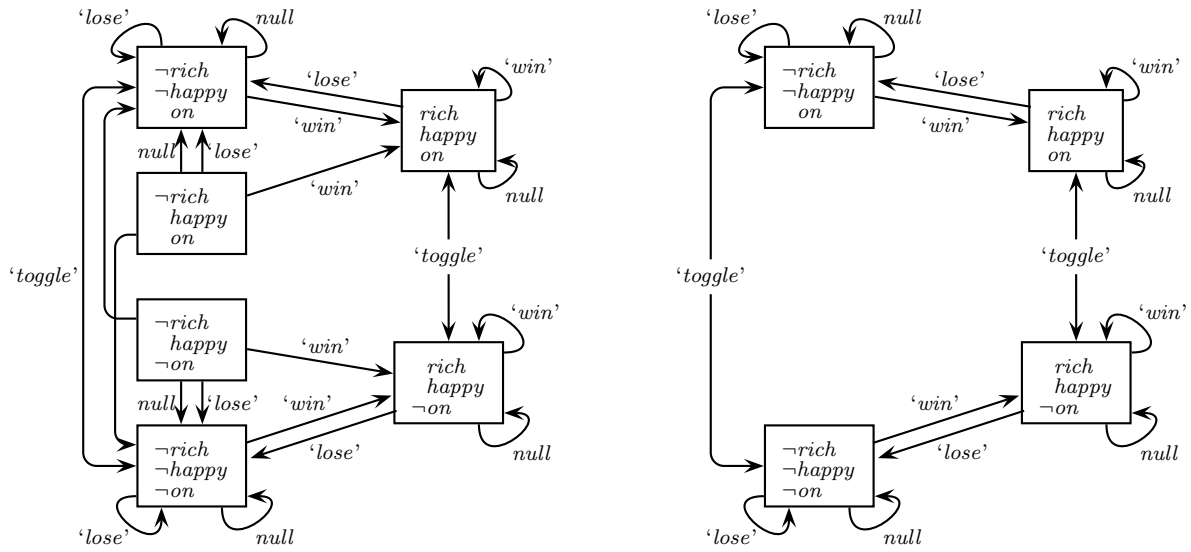
Suppose instead of inertial *happy* we add another static law

$$\mathsf{default}\ \neg happy \qquad (\text{i.e., } \neg happy\ \mathsf{if}\ \neg happy)$$

Now we get transitions to $\neg happy$ states, but still no transitions to *happy* states where *rich* does not hold. Also, some *null* and '*lose*' transitions have different effects, since *happy* goes to $\neg happy$ by default. This is illustrated in the leftmost of the two diagrams below.

Suppose we make *happy* exogenous (in effect adding another law default *happy* to default $\neg happy$)? Now the static law *happy* if *rich* is strong enough to eliminate unwanted states but the '*toggle*' and *null* transitions, which were deterministic, now become non-deterministic. (We omit the diagram for this case.)

Finally, suppose instead of default *happy*, we add the static law $\neg happy$ if $\neg rich$. Then there are only four states instead of six: states with $happy \wedge \neg rich$ are eliminated. This is shown in the rightmost diagram below.



Now suppose we make *happy* a statically determined fluent constant. Then there are only two states in the transition system: $\{rich, happy, on\}$ and $\{rich, happy, \neg on\}$. This is easy to check using the characterisation of a state, $s = T_{static}(s) \cup Simple(s)$:

s		$T_{static}(s) \cup$	$Simple(s)$	
$\{rich, happy, on\}$	$=$	$\{happy\} \cup$	$\{rich, on\}$	$\surd$
$\{rich, happy, \neg on\}$	$=$	$\{happy\} \cup$	$\{rich, \neg on\}$	$\surd$
$\{\neg rich, happy, on\}$	$\neq$	$\{\} \cup$	$\{\neg rich, on\}$	$\times$
$\{\neg rich, happy, \neg on\}$	$\neq$	$\{\} \cup$	$\{\neg rich, \neg on\}$	$\times$
$\{\neg rich, \neg happy, on\}$	$\neq$	$\{\} \cup$	$\{\neg rich, on\}$	$\times$
$\{\neg rich, \neg happy, \neg on\}$	$\neq$	$\{\} \cup$	$\{\neg rich, \neg on\}$	$\times$

(We omit the two interpretations satisfying $rich \wedge \neg happy$ since they are obviously not states.)

Now add another static law: default $\neg happy$ (i.e., $\neg happy$ if $\neg happy$):

s	$=$	$T_{static}(s)$	$\cup$	$Simple(s)$	
$\{rich, happy, on\}$	$=$	$\{happy\}$	$\cup$	$\{rich, on\}$	$\surd$
$\{rich, happy, \neg on\}$	$=$	$\{happy\}$	$\cup$	$\{rich, \neg on\}$	$\surd$
$\{\neg rich, happy, on\}$	$\neq$	$\{\}$	$\cup$	$\{\neg rich, on\}$	$\times$
$\{\neg rich, happy, \neg on\}$	$\neq$	$\{\}$	$\cup$	$\{\neg rich, \neg on\}$	$\times$
$\{\neg rich, \neg happy, on\}$	$=$	$\{\neg happy\}$	$\cup$	$\{\neg rich, on\}$	$\surd$
$\{\neg rich, \neg happy, \neg on\}$	$=$	$\{\neg happy\}$	$\cup$	$\{\neg rich, \neg on\}$	$\surd$

Now we have another two states in the transition system: we have also $\{\neg rich, \neg happy, on\}$ and $\{\neg rich, \neg happy, \neg on\}$. Notice that when *happy* is statically determined, the pair of static laws *happy* if *rich* and default $\neg happy$ is equivalent to the pair *happy* if *rich* and $\neg happy$ if $\neg rich$. Both say that *happy* is true in a state iff *rich* is true. (When *happy* is not statically determined, the two pairs are not equivalent, as the figure above demonstrates: we have default $\neg happy$ in the diagram on the left and $\neg happy$ if $\neg rich$ in the diagram on the right. In particular, default $\neg happy$ admits states that are eliminated by the other formulation.)

Finally, we cannot add inertial *happy* when *happy* is statically determined, since inertial declarations are fluent dynamic laws and cannot have statically determined fluent constants in their head.

4.7 Example (Winning the lottery)

Winning the lottery causes one to become (or remain) rich. Losing one's wallet causes one to become (or remain) not rich. A person who is rich is happy.

Signature: simple Boolean fluent constants *alive*, *rich*, *happy*; exogenous Boolean action constants *birth*, *death*, *win*, *lose*.

inertial *alive*
inertial *rich*
inertial *happy*

birth causes *alive*
nonexecutable *birth* if *alive*

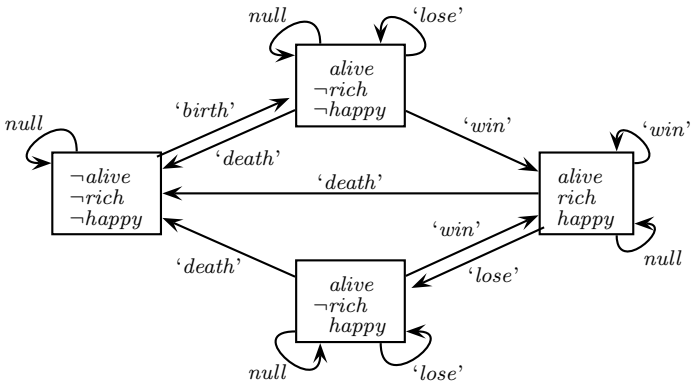
death causes $\neg alive$
nonexecutable *death* if $\neg alive$

win causes *rich*
nonexecutable *win* if $\neg alive$

lose causes $\neg rich$
nonexecutable *lose* if $\neg alive$

happy if *rich*
$\neg rich$ if $\neg alive$
$\neg happy$ if $\neg alive$

nonexecutable $birth \wedge death$
nonexecutable $birth \wedge win$
nonexecutable $birth \wedge lose$
nonexecutable $win \wedge lose$



The example is constructed partly to show how $\mathcal{C}+$ deals with indirect effects of actions (ramifications). Here we have two fluents, *happy* and *rich*, whose values are constrained by the presence of static laws. The first is not affected directly by any action, whereas the second one is.

Because of the static laws, there are only four states in the transition system and not $2^3 = 8$. Transition labels '*birth*', '*death*', '*win*', '*lose*' in the diagram are shorthand for the events $\{birth, \neg death, \neg win, \neg lose\}$, $\{\neg birth, death, \neg win, \neg lose\}$, $\{\neg birth, \neg death, win, \neg lose\}$, $\{\neg birth, \neg death, \neg win, lose\}$, respectively. The label *null* is shorthand for $\{\neg birth, \neg death, \neg win, \neg lose\}$.

Notice that as formulated here, the example allows for reincarnation: a person can be born, die, and be born again. The possibility of reincarnation can be eliminated easily enough, for example by adding another fluent constant *dead* together with a static law $\bot$ if $alive \wedge dead$; the simpler version with reincarnation is perfectly adequate for present purposes.

The diagram does not show transitions of type $death \wedge lose$ (i.e., transitions with label $\{\neg birth, death, \neg win, lose\}$). Their effects in this example are exactly the same as those of '*death*' transitions. There are no transitions of type $win \wedge death$: they would lead to states with $rich \wedge \neg alive$, and there can be no such states because of the static law $\bot$ if $rich \wedge \neg alive$. However, it seems unreasonable to insist that transitions of type $win \wedge death$ are non-executable. We can admit the possibility of $win \wedge death$ transitions by replacing the causes law for *win* by the following:

$$win \text{ causes } rich \text{ if } \neg death$$

or equivalently $win \wedge \neg death$ causes *rich*. (These statements are equivalent because they are shorthand for *rich* if $\top$ after $win \wedge \neg death$ and *rich* if $\top$ after $(win \wedge \neg death) \wedge \top$ respectively.) The effects of the '*win*' transitions are unchanged, but the transition system now contains transitions of type $win \wedge death$: their effects are exactly the same as those of '*death*' transitions and transitions of type $death \wedge lose$ transitions. More generally, we might want to say that winning the lottery makes a person rich *by default*, and is overriden in exceptional circumstances, such as dying as the lottery is being won.

Notice that *happy* is declared inertial, and so still persists even if one becomes not rich. That is why the '*lose*' transition from state $\{alive, rich, happy\}$ results in the state $\{alive, \neg rich, happy\}$. We could of course modify the action description so that *happy* is no longer inertial but defined to be true if and only if *rich* is true. Or we might prefer to make *happy* non-inertial and let the '*lose*' transition be non-deterministic. The interactions between these various adjustments are rather subtle, however, and not always immediately obvious. We will discuss some options below.

For illustration of the semantics, let us consider state $\{\neg alive, \neg rich, \neg happy\}$ and event '*birth*' (i.e., $\{birth, \neg death, \neg win, \neg lose\}$). There are no action dynamic laws to consider; the fluent dynamic laws that need to be considered are the law *birth* causes *alive*, the three inertial statements, and the nonexecutable statements.

- Is $(\{\neg alive, \neg rich, \neg happy\}, \text{'}birth\text{'}, \{alive, \neg rich, \neg happy\})$ a transition defined by the action description? It is, because $E(\{\neg alive, \neg rich, \neg happy\}, \text{'}birth\text{'}, \{alive, \neg rich, \neg happy\}) = \{alive, \neg rich, \neg happy\}$.

- Is $(\{\neg alive, \neg rich, \neg happy\}, `birth', \{alive, \neg rich, happy\})$ a transition defined by this action description? No, because $E(\{\neg alive, \neg rich, \neg happy\}, `birth', \{alive, \neg rich, happy\}) = \{alive, \neg rich\} \neq \{alive, \neg rich, happy\}$.
- Is there a transition of the form $(\{alive, \neg rich, \neg happy\}, `birth', s')$ defined by this action description? No: the law nonexecutable *birth* if *alive* means that $E(\{alive, \neg rich, \neg happy\}, `birth', s')$ must contain the element $\bot$, and no state s' can contain $\bot$.

Simple though it is, the example still raises a number of interesting questions. For example:

- Given the stated results of *birth* and *death*, the law nonexecutable $birth \wedge death$ is implied by the remaining parts of the action description, in the sense that it could be deleted without affecting the transition system that is defined. This is because we have the implied laws $(birth \wedge death)$ causes $(alive \wedge \neg alive)$, or equivalently $(birth \wedge death)$ causes $\bot$, which is just nonexecutable $birth \wedge death$. Similarly, in the original formulation, we have the implied law $(win \wedge lose)$ causes $(rich \wedge \neg rich)$, which gives nonexecutable $win \wedge lose$. But note that after the adjustment to take into account the possible overriding effects of *death*, we have only the implied law $(win \wedge lose)$ causes $(rich \wedge \neg rich)$ if $\neg death$, or equivalently, $(win \wedge \neg death \wedge lose)$ causes $(rich \wedge \neg rich)$, that is, nonexecutable $win \wedge \neg death \wedge lose$, or equivalently nonexecutable $win \wedge lose$ if $\neg death$. This clearly does *not* imply nonexecutable $win \wedge lose$.
- In the original formulation we have the implied law nonexecutable $win \wedge death$ because: we have the implied law $(win \wedge death)$ causes $(rich \wedge \neg alive)$, and there is no state in which $rich \wedge \neg alive$ holds because of the static law $\bot$ if $rich \wedge \neg alive$; so $(win \wedge death)$ causes $\bot$.
- The laws nonexecutable $birth \wedge win$ and nonexecutable $birth \wedge lose$ are also implied. We have *birth* causes $\bot$ if *alive* and *win* causes $\bot$ if $\neg alive$. So we can derive $(birth \wedge win)$ causes $\bot$ if *alive* and $(birth \wedge win)$ causes $\bot$ if $\neg alive$. Now we derive $(birth \wedge win)$ causes $\bot$ if $(alive \vee \neg alive)$. And similarly for nonexecutable $birth \wedge lose$.

Such questions are addressed in a separate paper (Sergot and Craven, 2005) on the logical properties of 'causal theories' and of the language $\mathcal{C}+$.

Instead of the static laws $\neg rich$ if $\neg alive$ and $\neg happy$ if $\neg alive$, suppose we have $\bot$ if $rich \wedge \neg alive$ and $\bot$ if $happy \wedge \neg alive$. What difference does it make, in this example, and in general? What about *alive* if *rich* and *alive* if *happy*?

First, notice that all three formulations are equivalent from the point of view of *states*. A state s satisfies the static law F if G iff it satisfies $\neg G$ if $\neg F$ iff it satisfies $\bot$ if $G \wedge \neg F$ iff $s \models G \rightarrow F$. But they are not equivalent from the point of view of transitions. If we replace $\neg happy$ if $\neg alive$ by either of *alive* if *happy* or $\bot$ if $happy \wedge \neg alive$, the only way that $\neg happy$ can be 'caused' is by inertia. Consequently, we eliminate all the *death* transitions ('*death*', and transitions of type $death \wedge lose$ and $death \wedge win$) from states in which *rich* holds. (Also, look at the causality reading of these rules.) If we replace $\neg rich$ if $\neg alive$ by either of *alive* if *rich* or $\bot$ if $rich \wedge \neg alive$, the argument is similar but a little more complicated. The only way that $\neg rich$ can be 'caused' is by a *lose* transition or by inertia. Consequently, the transitions '*death*' and transitions of type $death \wedge win$ become non-executable in the state $\{alive, rich, happy\}$.

There is one way in which we can use constraints $\bot$ if $rich \wedge \neg alive$ and $\bot$ if $happy \wedge \neg alive$ (or $alive$ if $rich$ and $alive$ if $happy$) without losing transitions. That is by adding a pair of extra fluent dynamic laws: either

$$death \text{ causes } \neg rich \quad \text{and} \quad death \text{ causes } \neg happy$$

or the weaker pair

$$death \text{ may cause } \neg rich \quad \text{and} \quad death \text{ may cause } \neg happy$$

The second pair seems preferable to the first, but neither is entirely satisfactory since they require all ramifications of *death* to be identified in advance and then modelled explictly using causal laws.

Statically determined fluents

To illustrate the difference between 'simple' and 'statically determined' fluents, suppose now that fluent constant *happy* is statically determined not simple. We have to remove the inertial *happy* declaration since this is not well formed when *happy* is statically determined. Suppose we have the static laws *happy* if *rich*, $\bot$ if $rich \wedge \neg alive$, and $\bot$ if $happy \wedge \neg alive$. There is only one state in the transition system defined by this action description, namely $\{alive, rich, happy\}$.

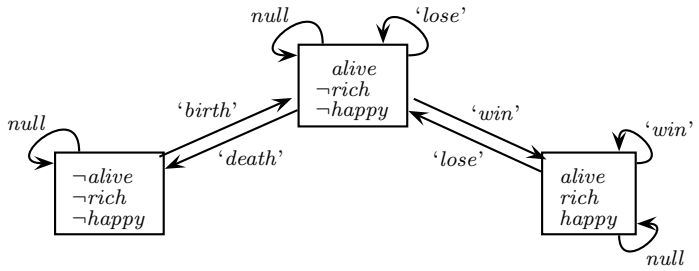
s	$=$	$T_{static}(s) \cup$	$Simple(s)$	
$\{\neg alive, \neg rich, \neg happy\}$	$\neq$	$\{\}$	$\cup \{\neg alive, \neg rich\}$	$\times$
$\{\ alive, \neg rich, \neg happy\}$	$\neq$	$\{\}$	$\cup \{alive, \neg rich\}$	$\times$
$\{\ alive, \neg rich, \ happy\}$	$\neq$	$\{\}$	$\cup \{alive, \neg rich\}$	$\times$
$\{\ alive, \ rich, \ happy\}$	$=$	$\{happy\} \cup$	$\{alive, rich\}$	$\surd$

(We omit the four interpretations satisfying $\neg alive \wedge rich$, $\neg alive \wedge happy$, or $rich \wedge \neg happy$ since they are obviously not states.)

If however we complete the definition of *happy* by adding another static law $\neg happy$ if $\neg happy$ (i.e., default $\neg happy$) then the only state eliminated from the original transition system is $\{alive, \neg rich, happy\}$, and we get three states.

s	$=$	$T_{static}(s) \cup$	$Simple(s)$	
$\{\neg alive, \neg rich, \neg happy\}$	$=$	$\{\neg happy\} \cup$	$\{\neg alive, \neg rich\}$	$\surd$
$\{\ alive, \neg rich, \neg happy\}$	$=$	$\{\neg happy\} \cup$	$\{alive, \neg rich\}$	$\surd$
$\{\ alive, \neg rich, \ happy\}$	$\neq$	$\{\}$ $\cup$	$\{alive, \neg rich\}$	$\times$
$\{\ alive, \ rich, \ happy\}$	$=$	$\{happy\} \cup$	$\{alive, rich\}$	$\surd$

We have the following transition system:



Notice that with the static laws as given above, there is no '$death'$ transition from $\{alive, rich, happy\}$ to $\{\neg alive, \neg rich, \neg happy\}$: there is nothing to 'cause' the transition from $rich$ to $\neg rich$. We can obtain the additional '$death'$ transition by adding, for example, the static law $\neg rich$ if $\neg alive$ (which implies, but is not implied by, $\bot$ if $rich \wedge \neg alive$).

4.8 Example (Going to work 1)

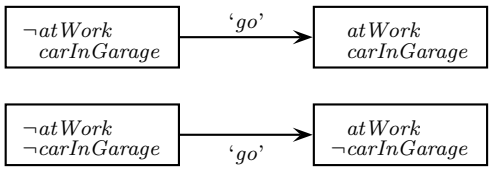
This is a modified version of an example used in (Giunchiglia and Lifschitz, 1998) to illustrate non-deterministic actions in $\mathcal{C}+$ and its precursor $\mathcal{C}$. There is a slightly more complicated version in (Giunchiglia et al., 2004). The simpler version is sufficient for the points we want to make here.

Let the (exogenous) Boolean action constant go represent 'Jack goes to work'. Jack can go to work by walking or, if his car is in his garage, he can drive. For simplicity, to simplify the diagrams, we ignore the possibility that Jack goes in the opposite direction.

The following action description

inertial $atWork$
inertial $carInGarage$
go causes $atWork$
nonexecutable go if $atWork$

makes 'go' deterministic in all states, as shown in the following diagram

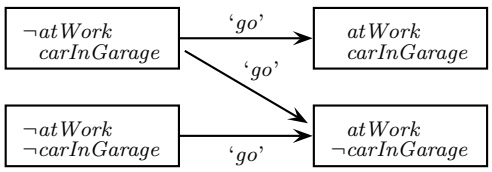


(Reflexive edges corresponding to the event $\{\neg go\}$ are not shown.)

However, what we expect (or want) is that 'go' is non-deterministic in those states where $carInGarage$ is true, because here Jack can either walk to work or drive and thereby move his car. To obtain this effect one adds another statement to the action description:

go may cause $\neg carInGarage$ if $carInGarage$

go may cause $\neg carInGarage$ if $carInGarage$ is an abbreviation for the fluent dynamic law $\neg carInGarage$ if $\neg carInGarage$ after $carInGarage \wedge go$. With the additional statement we obtain the following transition diagram ($\{\neg go\}$ events omitted):



4.9 Example (Going to work 2)

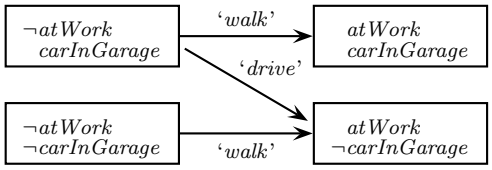
An alternative formulation of the preceding example would distinguish between walking to work and driving to work. Let us have two (exogenous) Boolean action constants *walk* and *drive* to represent walking and driving to work respectively. The action description

inertial $atWork$
inertial $carInGarage$

$walk$ causes $atWork$
$drive$ causes $atWork$
$drive$ causes $\neg carInGarage$ if $carInGarage$

nonexecutable $walk$ if $atWork$
nonexecutable $drive$ if $atWork$
nonexecutable $drive$ if $\neg carInGarage$
nonexecutable $walk \wedge drive$

defines the following transition system ($\{\neg walk, \neg drive\}$ omitted):



The first two causes laws could be replaced by the (equivalent) law:

$$(walk \vee drive) \text{ causes } atWork$$

We could also represent that *walk* and *drive* are both kinds of *go* by means of the action dynamic laws:

go if $walk$
go if $drive$
$\neg go$ if $\neg go$

The third law ($\neg go$ if $\neg go$) can be dropped when *go* is an exogenous action constant. When *go* is exogenous, we can also get the same effects by writing the pair of fluent dynamic laws:

nonexecutable $walk \wedge \neg go$
nonexecutable $drive \wedge \neg go$

We might also wish to add (in the absence of another kind of *go*, such as cycling):

$$\text{nonexecutable } go \wedge \neg walk \wedge \neg drive$$

This would not change the form of the transition system shown above except to replace transition labels '*walk*' and '*drive*' by $\{go, walk\}$ and $\{go, drive\}$ respectively.

Notice that the transition label $\{go, walk\}$, and indeed the action dynamic laws, cannot distinguish between two concurrent but unrelated actions *go* and *walk* and one action 'go by walking'. We return to this point when we discuss 'counts as' and act generation in Section 8 below.

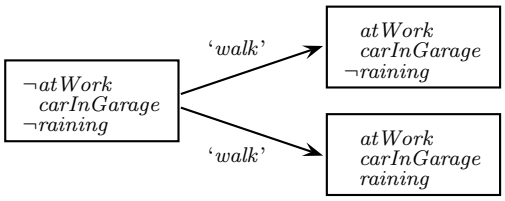
4.10 Example (Going to work 3)

There is another source of non-determinism: fluents which vary from state to state but are not 'caused' by any kind of action. An example is 'raining'. These are fluents which might properly be termed 'exogenous'.

For example: add to the signature a simple Boolean constant *raining*, and to the action description the pair of static laws:

$$\begin{array}{rl} \mathit{raining} & \textsf{if} \quad \mathit{raining} \\ \neg\mathit{raining} & \textsf{if} \quad \neg\mathit{raining} \end{array}$$

We obtain a transition system containing, for example, the fragment:



The pair of static laws for *raining* above may also be written more concisely in $\mathcal{C}+$ as:

$$\textsf{exogenous}\ \mathit{raining}$$

Recall that in general, for a fluent constant f, the abbreviation exogenous f stands for the set of static laws $f = v$ if $f = v$, for every $v \in \mathit{dom}(f)$.

5 Nonmonotonic causal theories

The language $\mathcal{C}+$ can be regarded as a higher-level notation for particular classes of theories in the formalism of 'nonmonotonic causal theories', and indeed this is how it is presented in (Giunchiglia et al., 2004). The formalism is also referred to as 'causal theories', and as 'the logic of causal explanation' in (Lifschitz, 1997). Turner (1999) has a more general formalism which he calls the 'logic of universal causation'.

Syntax A causal theory of signature σ is a set of expressions ('causal rules') of the form

$$F \Leftarrow G$$

where F and G are formulas of signature σ. A rule of this form is to be read as saying that F is 'caused' if G is true (which is not the same as saying that G 'causes' F).

Semantics Let Γ be a causal theory of signature σ and let X be an interpretation of its signature. The *reduct* Γ^X is the set of heads of rules of Γ whose bodies are satified by the interpretation X:

$$\Gamma^X =_{\text{def}} \{F \mid F \Leftarrow G \text{ is a rule in } \Gamma \text{ and } X \models G\}$$

X is a *model* of Γ iff X is the unique model of Γ^X, i.e., the unique interpretation of σ that satisfies all formulas in Γ^X.

The rationale is this. Γ^X is the set of formulas that are 'caused' or 'causally explained' according to the rules of Γ, under interpretation X. If Γ^X has no models, or more than one model, or a unique model different from X, then X is not considered to be a model of Γ. Γ is *consistent* or *satisfiable* if it has a model.

Definite clausal theories A causal theory Γ is *definite* if

- the head of every rule of Γ is an atom or $\bot$, and
- no atom is the head of infinitely many rules of Γ.

(Compare with the definition of definite action descriptions given earlier.) Recall that as defined earlier, when p is a Boolean constant, $\neg p$ is shorthand for the atom $p = \mathsf{f}$ and so covered by the definition.

5.1 Translation into (classical) propositional logic

Definite causal theories can be translated via the process of 'literal completion' into expressions of (classical) propositional logic. The process is analogous to the Clark (predicate) completion that provides the original semantics for negation by failure in logic programs.

The completion of a definite causal theory Γ, $comp(\Gamma)$, is defined as follows.

Consider a definite causal theory Γ of signature σ. For each non-trivial atom A the *completion formula* for A is the formula

$$A \leftrightarrow G_1 \vee \cdots \vee G_n$$

where $G_1, \ldots, G_n$ $(n \geq 0)$ are the bodies of the rules of Γ which have head A. The *completion* of Γ, $comp(\Gamma)$, is the set of formulas obtained by taking the completion formulas for all non-trivial atoms of σ along with the formula $\neg F$ for each rule of the form $\bot \Leftarrow F$ in Γ.

Notice:

1. The completion formula for any non-trivial atom A that is the head of no rule in Γ (the case $n = 0$) is $A \leftrightarrow \bot$.
2. For a *trivial* atom A nothing is included in the completion. Alternatively, we could add A to the completion. It would make no difference to the models of the completion but might perhaps make it clearer.
3. We are treating Boolean constants as a special case of two-valued constants, with $\neg p$ standing for $p = \mathsf{f}$ when p is a Boolean constant.

Proposition (Proposition 6 in (Giunchiglia et al., 2004)) *The models of a definite causal theory are precisely the models of its completion.*

Careful: the causal theory $\{p \Leftarrow q\}$ has no models. The completion of this theory is *not* $\{p \leftrightarrow q\}$, which has models, but $\{p \leftrightarrow q, \neg p \leftrightarrow \bot, q \leftrightarrow \bot, \neg q \leftrightarrow \bot\}$, which has no models.

As already observed, the task of finding a model of the completion $comp(\Gamma)$ of a definite causal theory Γ—and generally the task of finding a model of any set of formulas of a multi-valued signature—can be reduced to the task of finding a model of a set of classical propositional formulas.

6 Translation of $\mathcal{C}+$ to causal theories

For any action description D in $\mathcal{C}+$, and any non-negative integer m, it is possible to construct a causal theory Γ_m^D such that the models of Γ_m^D correspond to the paths of length m of the transition system defined by D. The language $\mathcal{C}+$ can thus be regarded as a higher-level notation for defining causal theories of a particular kind, and indeed this is exactly as it is presented in (Giunchiglia et al., 2004).

When the action signature of D is $(\sigma^{\mathrm{f}}, \sigma^{\mathrm{a}})$ the signature of Γ_m^D is obtained by time-stamping every fluent and action constant of D with non-negative integers between 0 and m: in the notation of Section 3.3 this is the signature $\sigma_m = \sigma_m^{\mathrm{f}} \cup \sigma_{m-1}^{\mathrm{a}}$, where $\sigma_m^{\mathrm{f}} = \sigma^{\mathrm{f}}[0] \cup \cdots \cup \sigma^{\mathrm{f}}[m]$ and $\sigma_m^{\mathrm{a}} = \sigma^{\mathrm{a}}[0] \cup \cdots \cup \sigma^{\mathrm{a}}[m{-}1]$. Note in particular that the signature σ_0 of Γ_0^D is $\sigma^{\mathrm{f}}[0]$ and the signature σ_1 of Γ_1^D is $\sigma^{\mathrm{f}}[0] \cup \sigma^{\mathrm{a}}[0] \cup \sigma^{\mathrm{f}}[1]$.

Given an action description D, and a non-negative integer m, the causal theory Γ_m^D is constructed as follows. For every static law F if G in D, include a causal rule of the form

$$F[i] \Leftarrow G[i] \qquad \text{for every } i \in 0\,..\,m.$$

For every fluent dynamic law F if G after ψ in D, include a causal rule of the form

$$F[i{+}1] \Leftarrow G[i{+}1] \wedge \psi[i] \qquad \text{for every } i \in 0\,..\,m{-}1.$$

For every action dynamic law α if ψ in D, include a causal rule of the form

$$\alpha[i] \Leftarrow \psi[i] \qquad \text{for every } i \in 0\,..\,m{-}1.$$

We also require the following 'exogeneity laws'. For every *simple* fluent constant f and every $v \in dom(f)$, include a causal rule:

$$f[0] = v \Leftarrow f[0] = v$$

And for every *exogenous* action constant a and every $v \in dom(a)$, include a causal rule:

$$a[i] = v \Leftarrow a[i] = v \qquad \text{for every } i \in 0\,..\,m{-}1.$$

As observed earlier, the presentation of $\mathcal{C}+$ in (Giunchiglia et al., 2004) does not treat the specification of exogenous action constants as part of the signature but instead requires statements of the form

exogenous a

to be included explicitly in the action description for every action constant a intended to be exogenous. exogenous a is shorthand for the fluent dynamic laws

$$a = v \text{ if } a = v, \quad \text{for all } v \in dom(a)$$

One can see that the translation to the causal theory Γ_m^D is the same whichever treatment of exogenous action constants is taken.

For convenience, we list here the translated forms of the most commonly used abbreviations of $\mathcal{C}+$:

default F	$F[i] \Leftarrow F[i]$
default F if G	$F[i] \Leftarrow F[i] \wedge G[i]$
inertial f	$f[i{+}1] = v \Leftarrow f[i{+}1] = v \wedge f[i] = v$, for all $v \in dom(f)$
inertial f if ψ	$f[i{+}1] = v \Leftarrow f[i{+}1] = v \wedge f[i] = v \wedge \psi[i]$, for all $v \in dom(f)$
α causes F	$F[i{+}1] \Leftarrow \alpha[i]$
α causes F if ψ	$F[i{+}1] \Leftarrow \alpha[i] \wedge \psi[i]$
nonexecutable α	$\bot \Leftarrow \top \wedge \alpha[i]$
nonexecutable α if ψ	$\bot \Leftarrow \top \wedge \alpha[i] \wedge \psi[i]$
α may cause F	$F[i{+}1] \Leftarrow F[i{+}1] \wedge \alpha[i]$
α may cause F if ψ	$F[i{+}1] \Leftarrow F[i{+}1] \wedge \alpha[i] \wedge \psi[i]$

Clearly any interpretation X of the signature σ_m of Γ_m^D can be written in the form

$$s_0[0] \cup \varepsilon_0[0] \cup s_1[1] \cup \cdots \cup s_{m-1}[m{-}1] \cup \varepsilon_{m-1}[m{-}1] \cup s_m[m]$$

where each $s_i[i]$ and $\varepsilon_i[i]$ is an interpretation of $\sigma^{\mathrm{f}}[i]$ and $\sigma^{\mathrm{a}}[i]$, respectively.

Now we relate models of causal theories Γ_m^D to the definitions of states and transitions of an action description D as given in Section 4.2. The second part of Theorem 9 below corresponds to Proposition 8 of (Giunchiglia et al., 2004). The statements and proofs are different, however, because in (Giunchiglia et al., 2004) states and transitions are defined in terms of models of Γ_0^D and Γ_1^D respectively, and not explicitly in terms of transition systems as in Section 4.2. The following result establishes that the definitions of Section 4.2 are equivalent to those of (Giunchiglia et al., 2004). The proofs are not difficult but they are rather long to present in detail and so we defer them to a separate section so as not to disrupt the flow of the presentation.

Theorem 9 *Let D be an action description with signature $(\sigma^{\mathrm{f}}, \sigma^{\mathrm{a}})$. An interpretation $s[0]$ of $\sigma^{\mathrm{f}}[0]$ is a model of Γ_0^D iff s is a state of D.*

An interpretation $X \in \mathrm{I}(\sigma_m)$ of the form $s_0[0] \cup \varepsilon_0[0] \cup \cdots \cup \varepsilon_{m-1}[m{-}1] \cup s_m[m]$ is a model of Γ_m^D iff $(s_0\, \varepsilon_0\, \cdots\, \varepsilon_{m-1}\, s_m)$ is a path of length $m \geq 0$ of the transition system defined by D.

In particular, Γ_1^D represents paths of length 1 of D, i.e., the transitions defined by the action description D. Γ_0^D represents the states of the transition system defined by D: an interpretation s of the fluent constants σ^{f} is a state of D precisely when $s[0]$ is a model of Γ_0^D.

Since action descriptions in $\mathcal{C}+$ can be regarded as abbreviations for causal theories of a certain kind (above), literal completions for laws of $\mathcal{C}+$ can be defined also. Action descriptions in $\mathcal{C}+$ are thereby translated into formulas of (classical) propositional logic, which in turn can be submitted to a standard

propositional satisfiability solver. Although this method is limited to action descriptions that are *definite*, this is not an important restriction in practical applications.

6.1 Proof of Theorem 9

First, consider the reduct $(\Gamma_0^D)^{s[0]}$. It consists of:

- formulas $F[0]$ for every static law F if G in D such that $s[0] \models_{\sigma^{\mathrm{f}}[0]} G[0]$, i.e., $F[0]$ for every $F \in T_{static}(s)$; we write $T_{static}(s)[0]$ for this set of formulas;
- atoms $f[0]=v$ for every simple fluent constant f such that $s[0] \models_{\sigma^{\mathrm{f}}[0]} f[0]=v$; this is the set $Simple(s[0])$.

Let s' be an interpretation of $\sigma^{\mathrm{f}}[0]$. It is easy to see that $s'[0] \models_{\sigma^{\mathrm{f}}[0]} (\Gamma_0^D)^{s[0]}$ iff $s'[0] \models_{\sigma^{\mathrm{f}}[0]} T_{static}(s)[0] \cup Simple(s[0])$ iff $s' \models T_{static}(s) \cup Simple(s)$.

And $s'[0]$ is the only model of $(\Gamma_0^D)^{s[0]}$ iff s is the only model of $T_{static}(s) \cup Simple(s)$ iff s is a state of D. □

Now let X be an interpretation of σ_m of the form

$$s_0[0] \cup \varepsilon_0[0] \cup \cdots \cup \varepsilon_{m-1}[m-1] \cup s_m[m]$$

First, consider the reduct $(\Gamma_m^D)^X$. It consists of:

- formulas $F[i]$ for every static law F if G in D such that $s_i[i] \models_{\sigma^{\mathrm{f}}[i]} G[i]$ for $0 \le i \le m$, i.e., $F[i]$ for every $F \in T_{static}(s_i)$ for $0 \le i \le m$; this is the set $T_{static}(s_0)[0] \cup \cdots \cup T_{static}(s_1)[0] \cup \cdots \cup T_{static}(s_m)[m]$;
- atoms $f[0]=v$ for every simple fluent constant f such that $s_0[0] \models_{\sigma^{\mathrm{f}}[0]} f[0]=v$, i.e., $Simple(s_0[0])$;
- formulas $F[i+1]$ for every fluent dynamic law F if G after ψ in D such that $s_{i+1}[i+1] \models_{\sigma^{\mathrm{f}}[i+1]} G[i+1]$ and $s_i[i] \cup \varepsilon_i[i] \models_{\sigma[i]} \psi[i]$ for $0 \le i < m$, i.e., $F[i+1]$ for every fluent dynamic law F if G after ψ in D such that $s_{i+1} \models G$ and $s_i \cup \varepsilon_i \models \psi$ for $0 \le i < m$; following the previous notation, this is the union of sets $E(s_i, \varepsilon_i, s_{i+1})[i+1]$ for $0 \le i < m$;
- formulas $\alpha[i]$ for every action dynamic law α if ψ in D such that $s_i[i] \cup \varepsilon_i[i] \models_{\sigma[i]} \psi[i]$ for $0 \le i < m$, i.e., $\alpha[i]$ for every action dynamic law α if ψ in D such that $s_i \cup \varepsilon_i \models \psi$ for $0 \le i < m$; this is the union of sets $A(\varepsilon_i, s_i)[i]$ for $0 \le i < m$;
- atoms $\alpha[i]=v$ for every exogenous action atom in σ^{a} such that $\varepsilon_i[i] \models_{\sigma^{\mathrm{a}}[i]} \alpha[i]=v$ for $0 \le i < m$, i.e., the union of sets $Exog(\varepsilon_i[i])$ for $0 \le i < m$.

Let X' be the interpretation $s'[0] \cup e'[0] \cup \cdots \cup e'[m-1] \cup s'[m]$ of σ_m.

$X' \models_{\sigma_m} (\Gamma_m^D)^X$ iff

$$
\begin{aligned}
s'[0] &\models_{\sigma^{\mathrm{f}}[0]} T_{static}(s_0)[0] \cup \mathit{Simple}(s_0[0]) \\
e'[0] &\models_{\sigma^{\mathrm{a}}[0]} A(\varepsilon_0, s_0)[0] \cup \mathit{Exog}(\varepsilon_0[0]) \\
s'[1] &\models_{\sigma^{\mathrm{f}}[1]} T_{static}(s_1)[1] \cup E(s_0, \varepsilon_0, s_1)[1] \\
&\vdots \\
s'[i] &\models_{\sigma^{\mathrm{f}}[i]} T_{static}(s_i)[i] \cup E(s_{i-1}, \varepsilon_{i-1}, s_i[i] \\
e'[i] &\models_{\sigma^{\mathrm{a}}[i]} A(\varepsilon_i, s_i)[i] \cup \mathit{Exog}(\varepsilon_i[i]) \\
s'[i{+}1] &\models_{\sigma^{\mathrm{f}}[i+1]} T_{static}(s_{i+1})[i{+}1] \cup E(s_i, \varepsilon_i, s_{i+1})[i{+}1] \\
&\vdots \\
e'[m{-}1] &\models_{\sigma^{\mathrm{a}}[m-1]} A(\varepsilon_{m-1}, s_{m-1})[m{-}1] \cup \mathit{Exog}(\varepsilon_{m-1}[m-1]) \\
s'[m] &\models_{\sigma^{\mathrm{f}}[m]} T_{static}(s_m)[m] \cup E(s_{m-1}, \varepsilon_{m-1}, s_m)[m]
\end{aligned}
$$

It follows that $X' \models_{\sigma_m} (\Gamma_m^D)^X$ iff

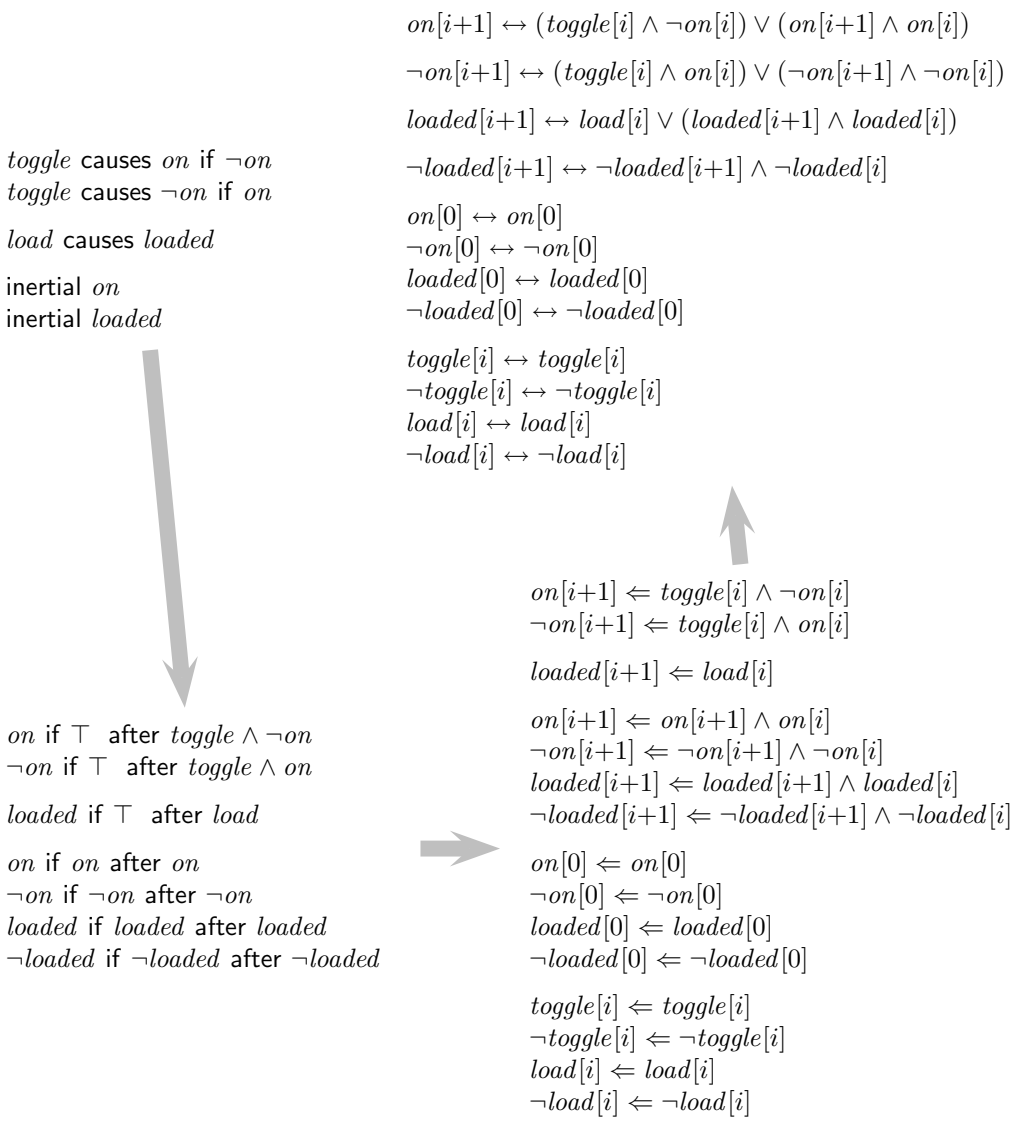
$$
\begin{aligned}
s'_0 &\models T_{static}(s_0) \cup \mathit{Simple}(s_0) \\
\varepsilon_0 &\models A(\varepsilon_0, s_0) \cup \mathit{Exog}(\varepsilon_0) \\
s'_1 &\models T_{static}(s_1) \cup E(s_0, \varepsilon_0, s_1) \\
&\vdots \\
s'_i &\models T_{static}(s_i) \cup E(s_{i-1}, \varepsilon_{i-1}, s_i) \\
\varepsilon_i &\models A(\varepsilon_i, s_i) \cup \mathit{Exog}(\varepsilon_i) \\
s'_{i+1} &\models T_{static}(s_{i+1}) \cup E(s_i, \varepsilon_i, s_{i+1}) \\
&\vdots \\
\varepsilon_{m-1} &\models A(\varepsilon_{m-1}, s_{m-1}) \cup \mathit{Exog}(\varepsilon_{m-1}) \\
s'_m &\models T_{static}(s_m) \cup E(s_{m-1}, \varepsilon_{m-1}, s_m)
\end{aligned}
$$

And X is the only model of $(\Gamma_m^D)^X$ iff

$$
\begin{aligned}
s'_0 &\;!\models T_{static}(s_0) \cup \mathit{Simple}(s_0) \\
\varepsilon_0 &\;!\models A(\varepsilon_0, s_0) \cup \mathit{Exog}(\varepsilon_0) \\
s'_1 &\;!\models T_{static}(s_1) \cup E(s_0, \varepsilon_0, s_1) \\
&\vdots \\
s'_i &\;!\models T_{static}(s_i) \cup E(s_{i-1}, \varepsilon_{i-1}, s_i) \\
\varepsilon_i &\;!\models A(\varepsilon_i, s_i) \cup \mathit{Exog}(\varepsilon_i) \\
s'_{i+1} &\;!\models T_{static}(s_{i+1}) \cup E(s_i, \varepsilon_i, s_{i+1}) \\
&\vdots \\
\varepsilon_{m-1} &\;!\models A(\varepsilon_{m-1}, s_{m-1}) \cup \mathit{Exog}(\varepsilon_{m-1}) \\
s'_m &\;!\models T_{static}(s_m) \cup E(s_{m-1}, \varepsilon_{m-1}, s_m)
\end{aligned}
$$

But by Proposition 5, these are precisely the conditions for $s_0\, \varepsilon_0 \cdots \varepsilon_{m-1}\, s_m$ to be a path of length m of D. □

6.2 Example



Note the introduction of the 'exogeneity laws' in the second step.

7 Computational tasks

$\mathcal{C}+$ is a language for defining labelled transition systems (of a certain kind). A wide variety of languages—'query languages'—can be interpreted on the transition systems so defined. These include simple propositional languages, as well as temporal logics such at CTL and LTL, and potentially many others. The query language supported by the Causal Calculator CCalc is the time-stamped language discussed in Section 3.3 for expressing properties of paths/runs of some

specified length m of a transition system. For an action description of signature $(\sigma^{\mathrm{f}}, \sigma^{\mathrm{a}})$ and non-negative integer m, this is the propositional language of signature $\sigma^{\mathrm{f}}_m \cup \sigma^{\mathrm{a}}_{m-1}$, which is also the signature of the causal theory Γ^D_m.

For example, for the 'lottery example' of Section 4.7, the query

$$\neg \mathit{alive}[0] \wedge \mathit{alive}[m] \wedge \mathit{happy}[m]$$

asks whether there is a path of length m in the transition system such that *alive* is false in the initial state while *alive* and *happy* are both true in state m.

For a definite action description D, a formula ψ of the query language can be evaluated by finding (classical) models of $\mathit{comp}(\Gamma^D_m)$ (and therefore paths of length m of the transition system defined by D) which are also models of ψ, i.e., finding models of $\mathit{comp}(\Gamma^D_m) \cup \{\psi\}$.

This then is the basic operation of the Causal Calculator CCalc. For a given action description D of signature $(\sigma^{\mathrm{f}}, \sigma^{\mathrm{a}})$, non-negative integer m, and query ψ of the time-stamped signature σ_m, CCalc

- performs the translation of D to Γ^D_m,
- constructs $\mathit{comp}(\Gamma^D_m)$,
- invokes a standard propositional sat-solver to find (classical) models of $\mathit{comp}(\Gamma^D_m) \cup \psi$, and then
- post-processes the sat-solver output to show the models obtained.

In practice, the first two steps may be combined into one, possibly with some additional optimisations to simplify the set of formulas passed to the sat-solver.

There is also a sub-language for specifying the action signature—sorts, variables, shorthand for various common forms of fluent, implementations of integer arithmetic, and so on—and a language for expressing particular common forms of queries. Details of these components are omitted here. They are not always well documented and vary slightly from one version of CCalc to another. See `http://www.cs.utexas.edu/users/tag/cc` for details of the current version of CCalc.

This general method covers a wide range of computational tasks. The account below follows a standard classification (prediction, 'postdiction', planning) current in the AI literature. CCalc has been applied successfully to a number of non-trivial benchmark examples in the temporal reasoning literature (see e.g. (Akman et al., 2004) and the CCalc website). In our own work we have used it to construct executable specifications of agent societies (see e.g. (Artikis et al., 2003a,b)).

Prediction Given an action description D of signature $(\sigma^{\mathrm{f}}, \sigma^{\mathrm{a}})$:

- Initially F holds.
- Partially specified events of type $\alpha_0, \alpha_1, \ldots, \alpha_k$ happen.
- Does G hold in state $k+1$?

In other words is there a path/run/history $s_0\, \varepsilon_0\, s_1 \cdots s_k\, \varepsilon_k\, s_{k+1}$ such that $s_0 \models F$, $\varepsilon_i \models \alpha_i$ for each $i \in 0..k$, and $s_{k+1} \models G$? F and G are formulas of σ^{f} and α_i are formulas of σ^{a}.

We want to know whether

$$comp(\Gamma^D_{k+1}) \models (F[0] \wedge \alpha_0[0] \wedge \alpha_1[1] \wedge \cdots \wedge \alpha_k[k] \rightarrow G[k+1])$$

We use a sat-solver to check whether

$$comp(\Gamma^D_{k+1}) \cup \{F[0] \wedge \alpha_0[0] \wedge \alpha_1[1] \wedge \cdots \wedge \alpha_k[k] \wedge \neg G[k+1]\}$$

is satisfiable.

A variant of the problem:

- Initially F holds.
- Partially specified events of type $\alpha_0, \alpha_1, \ldots, \alpha_k$ happen.
- Is it *possible* that G holds in state $k+1$? In other words, is there a possible run/path/history through the transition system such that G holds at its final state?

We check whether

$$comp(\Gamma^D_{k+1}) \cup \{F[0] \wedge \alpha_0[0] \wedge \alpha_1[1] \wedge \cdots \wedge \alpha_k[k] \wedge \neg G[k+1]\}$$

is satisfiable. If satisfiable, a propositional sat-solver will return all models.

'Postdiction' (stupid term)

- Partially specified events of type $\alpha_0, \alpha_1, \ldots, \alpha_k$ happen.
- G holds now.
- Does it follow that initially F?

We want to know whether

$$comp(\Gamma^D_{k+1}) \models (\alpha_0[0] \wedge \alpha_1[1] \wedge \cdots \wedge \alpha_k[k] \wedge G[k+1] \rightarrow F[0])$$

We check whether

$$comp(\Gamma^D_{k+1}) \cup \{\alpha_0[0] \wedge \alpha_1[1] \wedge \cdots \wedge \alpha_k[k] \wedge G[k+1] \wedge \neg F[0]\}$$

is satisfiable. And as before, checking whether

$$comp(\Gamma^D_{k+1}) \cup \{\alpha_0[0] \wedge \alpha_1[1] \wedge \cdots \wedge \alpha_k[k] \wedge G[k+1] \wedge F[0]\}$$

is satisfiable deals with the variant of the problem in which we want to know whether it is *possible* that initially F.

Temporal interpolation Prediction and 'postdiction' are both special cases of the general problem in which:

- Partially specified events of type $\alpha_0, \alpha_1, \ldots, \alpha_k$ happen.
- Certain combinations of fluents (partially specified states) hold at given times.

We want to determine what holds in each state, or what *possibly* holds in each state.

Planning

- Initially F.
- Goal: G.

Find the shortest sequence of fully specified actions (i.e., events, or transition labels) $\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_{k-1}$ such that there is a path/run/history $s_0\, \varepsilon_0\, s_1 \cdots s_{k-1}\, \varepsilon_{k-1}\, s_k$ in which $s_0 \models F$ and $s_k \models G$.

We try *consecutively* for $k = 0, 1, \ldots$ up to some specified maximum value m:

$$comp(\Gamma_k^D) \cup \{F[0] \wedge G[k]\} \quad \text{satisfiable ?}$$

The sat-solver returns all models, and these contain a representation of the plan: $\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_{k-1}$.

(The reference here is to events ε_i because we want fully specified actions in the plan. In the other problems, a formula α_i represents a partially specified event/transition.)

Notice that this formulation of planning is rather simplistic, in several respects. If any of the events in the plan $\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_{k-1}$ are non-deterministic, execution of the sequence of events $\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_{k-1}$ in the initial state may *possibly* result in the goal state but there is no guarantee that it will. Moreover, there is no distinction here between events that are effectively realisable and those that or not. For example, a possible 'plan' for becoming rich in the lottery example is to win the lottery. Possible refinements to deal with practical planning problems are outside the scope of this document.

Other possible problems

- Given a sequence of (partially specified) events $\alpha_0\, \alpha_1\, \ldots \alpha_k$ (no gaps), is this consistent with a given transition system (action description) D? This can be combined with partial information about these actions, and about some or all of the states. This is an instance of the temporal interpolation problem above.
- Given a sequence of (partially specified) events $\alpha_0\, \alpha_1\, \ldots \alpha_k$, but with possible gaps, is *this* consistent with a given transition system (action description) D? What are the complete (no gap) sequences of events?

  In principle this problem could also be solved, by iterating various sat-solver tests as in the planning problem to cover in some systematic fashion single gaps of one missing event, single gaps of j missing events, multiple gaps of single and j missing events, and so on, and so on, in all combinations. Clearly this method is not feasible without some further restrictions on possible gaps because of the (infinitely many) combinations to be considered.

Model checkers for temporal logic

A main attraction of the $\mathcal{C}+$ formalism compared to other action languages in the AI literature is that it has an explicit semantics in terms of transition systems, providing a bridge between AI formalisms and methods in other areas of computer science. One way of exploiting this link is by applying model checkers

for temporal logics to verify system properties of transition systems defined using the language $\mathcal{C}+$ (and its extensions $(\mathcal{C}+)^{++}$ to be presented later).

A small example of the use of CTL with an action description of $\mathcal{C}+$ (actually $(\mathcal{C}+)^{++}$ to be introduced in later sections) is sketched in (Sergot, 2005). Further development, both of $\mathcal{C}+$ as an input language to standard temporal logic model checkers, and of the use of a modified form of CCALC as a bounded model checker, is a topic of current investigation and will be covered in a separate paper.

Translation into (extended) logic programs

An alternative implementation route for $\mathcal{C}+$ is provided via a translation of (a subclass of) causal theories into (extended) logic programs (Giunchiglia et al., 2004, Section 7.2). We will not pursue this line of development here. The following summary is included for completeness.

Consider a causal theory whose rules are all of the form

$$L_0 \Leftarrow L_1 \wedge \cdots \wedge L_n \tag{4}$$

where $L_0, \ldots, L_n (n \geq 0)$ are literals (i.e., of the form A or $\neg A$ for A an atom of the signature of the theory).

Each such causal rule is translated to a clause of an (extended) logic program:

$$L_0 \leftarrow not\, \overline{L}_1, \ldots, not\, \overline{L}_n$$

where *not* is negation by failure and $\overline{L}_i$ stands for the literal complementary to L_i.

For $\mathcal{C}+$, a static law F if G thus translates to the logic program

$$F[i] \leftarrow not\, \neg G[i],$$

a law inertial p for Boolean constant p translates to the clauses

$$p[i{+}1] \leftarrow not\, \neg p[i{+}1], not\, \neg p[i] \tag{5}$$

$$\neg p[i{+}1] \leftarrow not\, p[i{+}1], not\, p[i], \tag{6}$$

and a law α causes F if ψ translates to the clause

$$F[i{+}1] \leftarrow not\, \neg \alpha[i], not\, \neg \psi[i].$$

Every causal theory of the form (4) can be translated to an equivalent (extended) logic program, in the following sense.

Proposition (Proposition 11 in (Giunchiglia et al., 2004)) *Let* Γ *be a causal theory whose rules have the form*

$$L_0 \Leftarrow L_1 \wedge \cdots \wedge L_n \qquad (n \geq 0)$$

where $L_0, \ldots, L_n$ *are literals. Let* $lp(\Gamma)$ *denote the (extended) logic program obtained by writing each such causal rule as a clause of the form:*

$$L_0 \leftarrow not\, \overline{L}_1, \ldots, not\, \overline{L}_n$$

An interpretation X of the signature of Γ is a model of the causal theory Γ iff X is an answer set for tbe logic program $lp(\Gamma)$.

Lifschitz and Turner (1999) present other, equivalent but more direct, translations of action descriptions into (extended) logic programs. For certain (common) special forms of action description, the translations of $\mathcal{C}+$ shown above can be simplified. A static law F if G becomes

$$F[i] \leftarrow G[i]$$

a law inertial p for Boolean constant p translates to the clauses

$$p[i{+}1] \leftarrow not\, \neg p[i{+}1], p[i] \tag{7}$$

$$\neg p[i{+}1] \leftarrow not\, p[i{+}1], \neg p[i], \tag{8}$$

and a law α causes F if ψ translates to the clause

$$F[i{+}1] \leftarrow \alpha[i], \psi[i].$$

The translation into logic programs provides another implementation route, employing answer set solvers such as SMODELS (Niemelä, 1999) and DLV (Eiter et al., 2001) in place of the satisfiability solvers used in CCALC. We shall not pursue this line of development.

We have a separate development, not presented here, that is attempting to develop an alternative translation to a different form of extended logic program with the same answer sets but with a different, 'event calculus' style, of computation (Craven and Sergot, 2003).

8 The language $(\mathcal{C}+)^+$: Action generation and 'counts as'

We now extend the language $\mathcal{C}+$ by adding expressions of the form α counts_as β for representing (a version of) the 'counts as' relation of (Jones and Sergot, 1996) or, essentially the same idea, what Goldman (1970) referred to as 'conventional generation'. We will call this extended language $(\mathcal{C}+)^+$. The counts_as atoms of $(\mathcal{C}+)^+$ can be seen as a distinguished type of fluent atom which has a special, fixed interpretation in an augmented kind of transition system. We want to be able to say that every transition of type α counts also (in specified circumstances) as a transition of type β. (Or more precisely: every transition of the type denoted by the action formula α counts (in specified circumstances) as a transition of the type denoted by action formula β.) The effects of such a transition are those that it has by virtue of being a transition of type α and those it has by virtue of being a transition of type β. This in turn provides us with a treatment of institutionalised power, as outlined in Section 2.

One simple possibility, which we shall reject, is to leave the notion of transition system itself unchanged, and simply require that the transition labels defined by a $(\mathcal{C}+)^+$ action description are closed under the generation/counts as rules. This is unsatisfactory, however. First, we would not be able to distinguish between α counts_as β and nonexecutable $\alpha \wedge \neg\beta$ ($\bot$ after $\alpha \wedge \neg\beta$). Second,

we want 'counts as' to be a semantic feature of a transition system and not just a syntactic feature of the $(\mathcal{C}+)^+$ language. And third, we want a *dynamic* notion of 'counts as' that can change under the effects of actions, so that we can say in the $(\mathcal{C}+)^+$ language, for example, that α' causes $(\alpha$ counts_as $\beta)$, or that α' causes $\mathsf{pow}(X, \alpha)$ where $\mathsf{pow}(X, \alpha)$ is an abbreviation for special forms of counts_as expressions. So we are left with the task of augmenting transition systems in one way or another.

We will present the development in two stages: first, a simple form of augmented transition system with a *static* notion of 'counts as' to introduce the basic idea, and then a more complicated structure to support the dynamic notion. We will then discuss how action descriptions of $(\mathcal{C}+)^+$ can be translated to $\mathcal{C}+$ and causal theories.

A more general version of the whole account, allowing arbitrary formulas of signature $\sigma^{\mathrm{f}} \cup \sigma^{\mathrm{a}}$ in counts_as expressions and not just action formulas, could also be constructed along similar lines but will not be explored here.

8.1 Preliminaries

We want to be able to say that, under specified conditions, every transition (s, ε, s') of type X counts, in institution $\mathcal{I}$, as a transition of type Y. First we must decide on what to take as the transition types. One obvious candidate is the subsets $\wp(R)$ of the set R of transitions. We would then define a counts as relation $C_{\mathcal{I}} \subseteq \wp(R) \times \wp(R)$. However, this is much more general than we need. It would allow, for instance, $(s_1, \varepsilon_1, s_1')$ counts as $(s_2, \varepsilon_2, s_2')$ even when $s_1 \neq s_1'$ and/or $s_2 \neq s_2'$. This is meaningful, but is much more general than we need.

Instead we take as types of transition the subsets $\wp(\mathbf{A})$ of the set $\mathbf{A}$ of transition labels. The transition (s, ε, s') is of type X $(X \subseteq \mathbf{A})$ when $\varepsilon \in X$. Since we want to use action formulas to denote types of transitions, we again consider the special case where transition labels are interpretations of σ^{a} (as opposed to specifying a separate valuation function for action constants of σ^{a}). For action formula α and a given transition system, let $\|\alpha\|$ denote the set of transition labels/events that satisfy α:

$$\|\alpha\| =_{\mathrm{def}} \{\varepsilon \in \mathrm{I}(\sigma^{\mathrm{a}}) \mid \varepsilon \models \alpha\}$$

As in previous sections, we will say that a transition is of type α when we mean that the transition is of type $\|\alpha\|$. Notice that if action formulas α and β are equivalent in the signature σ^{a} $(\alpha \equiv_{\sigma^{\mathrm{a}}} \beta)$ then $\|\alpha\| = \|\beta\|$.

Now we add to the labelled transition system an extra component

$$C_{\mathcal{I}} \subseteq S \times \wp(\mathbf{A}) \times \wp(\mathbf{A})$$

$C_{\mathcal{I}}(s, X, Y)$ represents that any transition $(s,\, e, s')$ of type X counts, in institution $\mathcal{I}$, as a transition of type Y. In principle there could be several such $C_{\mathcal{I}}$ components, one for each institution $\mathcal{I}$ that we wish to model. However, there are some further points of detail to be settled concerning the structure of states when there are multiple institutions, and to avoid those complications we stick to the case of a single institution for now (i.e., a single 'counts as' relation $C_{\mathcal{I}}$).

One might also consider a four-argument 'counts as' relation

$$C_{\mathcal{I}} \subseteq S \times \wp(\mathbf{A}) \times \wp(\mathbf{A}) \times S$$

but this is also more general than we need. It would only make a difference in the case of non-deterministic transitions ε where X counts as Y depends on the final state as well as the initial state of the transition. This is so rare it is not worth the extra complications.

Now the basic semantic structure, which we will call a 'statically augmented transition system', or 'static ATS' for short, is a structure of the form

$$\langle S, \mathbf{A}, R, C_{\mathcal{I}} \rangle$$

where S, $\mathbf{A}$, and R are the states, transition labels/events, and transitions of a labelled transition system, as usual, and $C_{\mathcal{I}}$ is the 'counts as' relation, $C_{\mathcal{I}} \subseteq S \times \wp(\mathbf{A}) \times \wp(\mathbf{A})$.

We further require that $C_{\mathcal{I}}$ satisfies the following condition, for all states $s \in S$ and all transition types $X, Y \subseteq \mathbf{A}$:

$$\text{if } C_{\mathcal{I}}(s, X, Y) \text{ then for any } (s, \varepsilon, s') \in R, \text{ if } \varepsilon \in X \text{ then } \varepsilon \in Y \qquad (\mathrm{C}'_{\varepsilon})$$

Condition $(\mathrm{C}'_{\varepsilon})$ captures the key idea that any transition of type X is also a transition of type Y. It can also be expressed in the form

$$\text{if } C_{\mathcal{I}}(s, X, Y) \text{ then } \tau_R(s, X) \subseteq \tau_R(s, Y)$$

where $\tau_R(s, X) =_{\text{def}} \{(s, \varepsilon, s') \in R \mid \varepsilon \in X\}$ denotes the set of transitions of type X that are executable in state s. Note that we do *not* want the converse of $(\mathrm{C}'_{\varepsilon})$: even if $\varepsilon \in X$ implies $\varepsilon \in Y$ for all $(s, \varepsilon, s') \in R$ we do not necessarily want to conclude that $C_{\mathcal{I}}(s, X, Y)$. There may be other rules operative in institution $\mathcal{I}$ (such as mere coincidence, and other kinds of causal and logical connections) which mean that all transitions of type X are also transitions of type Y, and we do not want to include these other connections as instances of the 'counts as' variety (Cf. (Jones and Sergot, 1996)). We will make some further comments about condition $(\mathrm{C}'_{\varepsilon})$ in Section 8.10.

Note that transition types and the counts as relation $C_{\mathcal{I}}$ are defined here in terms of states and transition labels/events, and are independent of the actual transition relation R.

The intention is this. We have some query language $\mathcal{L}_Q$ of signature $\sigma^{\mathrm{f}} \cup \sigma^{\mathrm{a}}$, or possibly its time-stamped version, of signature $\sigma^{\mathrm{f}}[0] \cup \sigma^{\mathrm{a}}[0] \cup \cdots \cup \sigma^{\mathrm{a}}[m-1] \cup \sigma^{\mathrm{f}}[m]$ for some non-negative integer m. This language will have an additional binary connective, say $\mathsf{counts\_as}_{\mathcal{I}}$, allowing expressions of the form $\alpha \,\mathsf{counts\_as}_{\mathcal{I}}\, \beta$ for α and β belonging to some designated set of action formulas, or $\alpha \,\mathsf{counts\_as}_{\mathcal{I}}[i]\, \beta$ in the time-stamped language. This $\mathsf{counts\_as}_{\mathcal{I}}$ connective is intended to be the analogue of the 'counts as' conditional of (Jones and Sergot, 1996) but restricted here to expressing 'counts as' relations between (designated pairs of) action formulas rather than formulas in general. Atoms of the language $\mathcal{L}_Q$ will be interpreted on states and transition labels as usual. $\mathsf{counts\_as}_{\mathcal{I}}$ expressions will be evaluated on states as follows:

$$s \models \alpha \,\mathsf{counts\_as}_{\mathcal{I}}\, \beta \quad \text{iff} \quad (s, \|\alpha\|, \|\beta\|) \in C_{\mathcal{I}}$$

or if $\mathcal{L}_Q$ is a time-stamped language evaluated on paths, then

$$\pi \models \ \alpha \,\mathsf{counts\_as}_{\mathcal{I}}[i]\, \beta \quad \text{iff} \quad (\pi_s[i], \|\alpha\|, \|\beta\|) \in C_{\mathcal{I}}$$

where $\pi_s[i]$ denotes the ith state of the path π, i.e., $\pi_s[i] = s_i$ when $\pi = s_0\, \varepsilon_0\, s_1\, \varepsilon_1\, \ldots\, \varepsilon_{i-1}\, s_i\, \ldots$.

(We are using different symbols $\mathsf{counts\_as}_{\mathcal{I}}$ and $\mathsf{counts\_as}$ so as not to confuse expressions of the query language with expressions of the $(\mathcal{C}+)^+$ action descriptions used to define an ATS. Informally, they are intended to represent the same concept. It will be convenient to keep the distinction when we discuss implementation options and translations of $(\mathcal{C}+)^+$ to causal theories in Section 8.9 below.)

Augmented action signatures We will again consider the special case in which states are interpretations of some set σ^{f} of fluent constants and transition labels/events are the interpretations of a set σ^{a} of action constants. We add a (disjoint) set σ^{γ} of Boolean constants of the form $\alpha\,\mathsf{counts\_as}\,\beta$ for α and β belonging to some designated set of action formulas of σ^{a}. We do not necessarily want to allow arbitrary formulas of α and β of σ^{a} in $\alpha\,\mathsf{counts\_as}\,\beta$ expressions. σ^{γ} allows these restrictions to be specified if desired. $(\sigma^{\mathrm{f}}, \sigma^{\gamma}, \sigma^{\mathrm{a}})$ is an *augmented action signature*. We will say that constants $\alpha\,\mathsf{counts\_as}\,\beta$ and $\alpha'\,\mathsf{counts\_as}\,\beta'$ of σ^{γ} are σ^{a}-equivalent when $\alpha \equiv_{\sigma^{\mathrm{a}}} \alpha'$ and $\beta \equiv_{\sigma^{\mathrm{a}}} \beta'$. Notice that since $\top$ and $\bot$ are not action formulas (they are 0-ary connectives) there are no constants of the form $\alpha\,\mathsf{counts\_as}\,\bot$ or $\top\,\mathsf{counts\_as}\,\beta$. As for the query language $\mathcal{L}_Q$, $\alpha\,\mathsf{counts\_as}_{\mathcal{I}}\,\beta$ is a well-formed expression of $\mathcal{L}_Q$ when $\alpha\,\mathsf{counts\_as}\,\beta$ is a constant of σ^{γ} (and likewise for the time-stamped query language).

Definition 10 (Static ATS) *A* statically augmented transition system (static ATS) *of signature* $(\sigma^{\mathrm{f}}, \sigma^{\gamma}, \sigma^{\mathrm{a}})$ *is a structure of the form*

$$\langle \sigma^{\mathrm{f}}, \sigma^{\gamma}, S, \mathrm{I}(\sigma^{\mathrm{a}}), R, C_{\mathcal{I}} \rangle$$

where

- $S \subseteq \mathrm{I}(\sigma^{\mathrm{f}})$ *is a set of states,*
- $\mathrm{I}(\sigma^{\mathrm{a}})$ *is the set of transition labels/events,*
- R *is the set of labelled transitions,* $R \subseteq S \times \mathrm{I}(\sigma^{\mathrm{a}}) \times S$*,*
- $C_{\mathcal{I}}$ *is the 'counts as' relation,* $C_{\mathcal{I}} \subseteq S \times \wp(\mathrm{I}(\sigma^{\mathrm{a}})) \times \wp(\mathrm{I}(\sigma^{\mathrm{a}}))$*,*

and, for all states $s \in S$ *and all transition types* $X, Y \subseteq \mathrm{I}(\sigma^{\mathrm{a}})$*:*

$$\text{if } C_{\mathcal{I}}(s, X, Y) \text{ then for any } (s, \varepsilon, s') \in R, \text{ if } \varepsilon \in X \text{ then } \varepsilon \in Y \qquad (\mathrm{C}'_{\varepsilon})$$

Remark (unsupported institutional actions) When $\alpha\,\mathsf{counts\_as}\,\beta$, β represents an institutional action (as opposed to a 'brute' or 'basic' action). α may represent an institutional action or a brute/basic action. In the present version, we have chosen not to make any explicit distinction between brute/basic and institutional actions in the semantics. Further, we have chosen not to eliminate, in the semantics, the possibility of *unsupported* institutional actions. β is an unsupported institutional action in a transition (s, ε, s') when $\alpha\,\mathsf{counts\_as}\,\beta$ is a

constant in σ^γ (so β represents an institutional action), $\varepsilon \models \beta$, but $\varepsilon \not\models \alpha'$ for any constant α' $\mathsf{counts\_as}$ β in σ^γ. For example, if *wave_flag* $\mathsf{counts\_as}$ *start_race* is a constant in σ^γ, and there is no other means of performing *start_race* besides *wave_flag* specified in σ^γ, then *start_race* is an unsupported institutional action in any transition of type *start_race* $\wedge \neg$*wave_flag*. It is a moot point whether unsupported institutional actions are meaningful given the intended informal reading. We have chosen not to eliminate them in the semantics (though this could be done easily) because unsupported institutional actions can be eliminated straightforwardly as desired when the action description is formulated, as will be illustrated by an example later. Not building this into the semantics gives more flexibility. We return to this point later in Section 8.8.

The treatment of 'counts as' presented here is guided by the account of 'counts as' conditionals and institutionalised power in (Jones and Sergot, 1996). A direct comparison is not possible because the semantic frameworks of the two accounts are very different, and because the dynamic setting adoped here raises a number of further considerations outside the scope of (Jones and Sergot, 1996). We limit ourselves to some brief remarks and some comments on the related concept of conventional generation of Goldman (1970).

The semantic counterpart of $C_\mathcal{I}$ in (Jones and Sergot, 1996) is transitive, at least in the absence of convincing counter-examples to the contrary. This suggests that we might strengthen the $C_\mathcal{I}$ relation, for example by imposing the following additional constraint, for all transition types $X, Y, Z \subseteq \mathbf{A}$ and all states s of the transition system:

$$\text{if } C_\mathcal{I}(s, X, Y) \text{ and } C_\mathcal{I}(s, Y, Z) \text{ then } C_\mathcal{I}(s, X, Z) \qquad (\mathrm{C}_{\mathrm{trans}})$$

This would mean that all states s have the property that

$$s \models \ (\alpha \,\mathsf{counts\_as}_\mathcal{I}\, \beta \wedge \beta \,\mathsf{counts\_as}_\mathcal{I}\, \gamma) \rightarrow \alpha \,\mathsf{counts\_as}_\mathcal{I}\, \gamma$$

though only when $\alpha\,\mathsf{counts\_as}\,\beta$, $\beta\,\mathsf{counts\_as}\,\gamma$, and $\alpha\,\mathsf{counts\_as}\,\gamma$ are all constants of σ^γ.

The possibility that 'counts as' could also be reflexive–that is, in the present context, that the following constraint could also be imposed

$$C_\mathcal{I}(s, X, X)$$

for all states s and all transition types $X \subseteq \mathbf{A}$–is explicitly rejected in (Jones and Sergot, 1996). The reason is simply that 'counts as' is not a mere form of classification, but is intended to convey the idea that certain kinds of action, or states of affairs more generally, have a special conventional significance within an institution. The action of opening a window, say, might have no special conventional significance within an institution: we would then not wish to say that opening a window 'counts as' opening a window. A reflexivity constraint could be imposed, but only in a restricted form, for special categories of transition types X, those corresponding to actions of conventional significance in institution $\mathcal{I}$. How are these special categories of transition types are to be characterised? This is not a question we will attempt to address in this paper. For present purposes we can say that X is a transition type in this special category when $X = \|\alpha\|$ or $X = \|\beta\|$ for some constant α $\mathsf{counts\_as}$ β in σ^γ. For

ease of reference, let us write $\mathbf{A}_{\mathcal{I}}$ for the transition types of special conventional significance in institution $\mathcal{I}$, as represented by the signature σ^{γ}: $\mathbf{A}_{\mathcal{I}} =_{\text{def}} \{X \subseteq \mathbf{A} \mid X = \|\alpha\| \text{ or } X = \|\beta\| \text{ for some constant } \alpha \,\mathsf{counts\_as}\, \beta \text{ in } \sigma^{\gamma}\}$. A suitably qualified reflexivity constraint would take the form

$$C_{\mathcal{I}}(s, X, X) \text{ if } X \subseteq \mathbf{A}_{\mathcal{I}} \qquad (\mathrm{C}_{\text{refl}})$$

Note that relying on the signature σ^{γ} in this way finesses the practical consequences of deciding what form of reflexivity constraint to adopt. The constraint (C_{refl}) yields the property that all states s satisfy

$$s \models \alpha \,\mathsf{counts\_as}_{\mathcal{I}}\, \alpha$$

but only when $\alpha \,\mathsf{counts\_as}\, \alpha$ is specified as a constant of σ^{γ}.

Jones and Sergot (1996) comment briefly on the similarities and some of the differences between their 'counts as' conditional and Goldman's notion of conventional generation. Let $G_I(s, X, Y)$ represent that a transition of type X (under conditions s) conventionally generates, in the norm/rule system (institution) $\mathcal{I}$, a transition of type Y. This move already deserves some comment [2]. Goldman's act generation is a relationship between act *tokens* not act *types*. So, for example, a particular instance of Jim's hammering a nail into a wall might 'causally generate' in Goldman's terminology a particular instance of Jim's waking Mary, but that would not be a relationship between all instances of Jim's hammering a nail into a wall and all instances of Jim's waking Mary. Conventional generation is different because norms/conventions tend to be general rather than specific, and so it is often meaningful to speak about conventional generation as a relationship between act *types*, as we do here.

Goldman argues that conventional generation, like other forms of act generation, is irreflexive and transitive. So let us impose the following constraints, for all states s in S and all subsets X, Y of $\mathbf{A}$:

- $(s, X, X) \notin G_{\mathcal{I}}$
- if $G_{\mathcal{I}}(s, X, Y)$ and $G_{\mathcal{I}}(s, Y, Z)$ then $G_{\mathcal{I}}(s, X, Z)$

We see that $G_{\mathcal{I}}(s, \cdot, \cdot)$ (or more precisely, the projection $G_{\mathcal{I}}(s, \cdot, \cdot)$ for fixed s) is a strict ordering on transition types. The natural interpretation is to see the relationship between 'counts as' $C_{\mathcal{I}}$ and Goldman's 'conventional generation' $G_{\mathcal{I}}$ as analogous to that between a partial order and its corresponding strict partial order, i.e., to expect that the following should hold: $C_{\mathcal{I}}(s, X, Y)$ iff either $G_{\mathcal{I}}(s, X, Y)$ or $X = Y$, for all states s and all transition types $X, Y \subseteq \mathbf{A}_{\mathcal{I}}$. $C_{\mathcal{I}}$ then comes out to be anti-symmetric, which suggests imposing the following further constraint on $C_{\mathcal{I}}$, for all states s in S and all subsets X, Y of $\mathbf{A}_{\mathcal{I}}$:

$$\text{if } C_{\mathcal{I}}(s, X, Y) \text{ and } C_{\mathcal{I}}(s, Y, X) \text{ then } X = Y \qquad (\mathrm{C}_{\text{a-symm}})$$

The 'counts as' relation $C_{\mathcal{I}}$ (or rather, each projection $C_{\mathcal{I}}(s, \cdot, \cdot)$ for fixed s) is a partial ordering, not on the set of all transition types, but on the set $\mathbf{A}_{\mathcal{I}}$ of transition types with special conventional significance in institution $\mathcal{I}$. The non-minimal elements of this ordering correspond to institutional actions, and

[2] The author is grateful to Andreas Herzig for a helpful reminder of this point.

the minimal ones to what we might call 'basic' actions. (We prefer the term 'basic' actions to Searle's 'brute' actions because many examples, such as the auction protocol presented in Section 8.7 below, can be usefully formalised at the level of basic actions such as 'signal a bid' without necessarily specifying in detail the 'brute' actions (physical actions, the transmission of specific forms of electronic messages, and so on) by means of which the making of a bid is actually signalled in the auction.)

Although this seems quite plausible, we shall not explore these possibilities in detail here. Since 'counts as' and conventional generation appear inter-definable (at least in principle, even if not exactly as suggested above), we shall fix on the 'counts as' reading $C_{\mathcal{I}}$ from now on. Moreover, we will not adopt the constraints ($\mathrm{C}_{\mathrm{refl}}$), ($\mathrm{C}_{\mathrm{trans}}$), ($\mathrm{C}_{\text{a-symm}}$) as fixed features of the semantics. Although it is tempting to do so, the practical significance of the reflexivity constraint ($\mathrm{C}_{\mathrm{refl}}$) is negligible, while the transitivity constraint is problematic to express in the formalism of causal theories. There are moreover alternative formulations, particularly of transitivity, that could also be considered. We defer further discussion until Section 8.9 below.

Finally, given the intended reading of counts_as constants, it would be natural to impose the additional restriction that the 'counts as' signature σ^{γ} must be cycle-free, in the sense that it must contain no constants α_0 counts_as α_1, $\alpha_1$counts_as$\alpha_2, \ldots, \alpha_{n-1}$counts_as$\alpha_n$ ($n \geq 1$) where $\alpha_n \equiv_{\sigma^{\mathrm{a}}} \alpha_0$, and $\alpha_i \equiv_{\sigma^{\mathrm{a}}} \alpha_{i+1}$ for $0 \leq i < n$. Again, this seems reasonable but there are some further points of detail to resolve, and we make no such assumption about σ^{γ} in the version presented in this paper. All the examples that will be discussed, however, will have σ^{γ} cycle-free in the sense described above.

8.2 Augmented transition systems

The 'static ATS' structure defined in the previous section is too simple for what we want. It provides only a *static* 'counts as' relation between transition types, whereas we want to support a dynamic version, that is, one in which transitions can change the $C_{\mathcal{I}}$ relation whether or not they also change the values of fluents. We need a more complicated structure. One possibility is to extend the static ATS structure with another transition relation, this time between 'counts as' relations: the tuple $(C_{\mathcal{I}}, \varepsilon, C'_{\mathcal{I}})$ would represent that a transition with label $\varepsilon \in \mathbf{A}$ transforms the 'counts as' relation from $C_{\mathcal{I}}$ to $C'_{\mathcal{I}}$. We would then need to formulate suitable constraints to ensure that transitions between states and transitions between $C_{\mathcal{I}}$ relations are coherent in some appropriate sense. Instead of that, we can get the same effect by defining a structure with a *single* transition relation, but one defined between *augmented states* $(s_{\mathrm{f}}, \gamma_{\mathcal{I}})$, each of which is characterised by the values s_{f} of the fluent constants and by a set $\gamma_{\mathcal{I}}$ of pairs (X, Y) specifying that in this augmented state a transition of type X counts as a transition of type Y. The condition corresponding to ($\mathrm{C}'_{\varepsilon}$) is then:

if $((s_{\mathrm{f}}, \gamma_{\mathcal{I}}), \varepsilon, (s'_{\mathrm{f}}, \gamma'_{\mathcal{I}})) \in R$, then for any $(X, Y) \in \gamma_{\mathcal{I}}$, if $\varepsilon \in X$ then $\varepsilon \in Y$

As usual, we will concentrate on the special case where states (now augmented states) are interpretations of the action signature. In an augmented action signature $(\sigma^{\mathrm{f}}, \sigma^{\gamma}, \sigma^{\mathrm{a}})$, σ^{f} and σ^{γ} are disjoint, and every augmented state s can

be represented by an interpretation of $\sigma^{\mathrm{f}} \cup \sigma^{\gamma}$, i.e., in the form $s_{\mathrm{f}} \cup \gamma$ where s_{f} is an interpretation of the fluent constants σ^{f} and γ is an interpretation of the 'counts as' constants σ^{γ}. When interpretations are represented by the set of atoms that they satisfy, s_{f} is a set of fluent atoms and each γ will be a set of $\mathsf{counts\_as}$ atoms. We need one further restriction: since $\|\alpha\| = \|\beta\|$ when $\alpha \equiv_{\sigma^{\mathrm{a}}} \beta$, the interpretation s must agree on the value of all σ^{a}-equivalent pairs of $\mathsf{counts\_as}$ constants in σ^{γ}; s (and therefore γ) must satisfy the condition $s \models (\alpha\,\mathsf{counts\_as}\,\beta \leftrightarrow \alpha'\,\mathsf{counts\_as}\,\beta')$ for all (distinct) pairs of constants $\alpha\,\mathsf{counts\_as}\,\beta$, $\alpha'\,\mathsf{counts\_as}\,\beta'$ in σ^{γ} such that $\alpha \equiv_{\sigma^{\mathrm{a}}} \alpha'$ and $\beta \equiv_{\sigma^{\mathrm{a}}} \beta'$.

As a small point of detail, since an augmented state is characterised by the values of the σ^{f} and σ^{γ} constants, we can allow the augmented action signature $(\sigma^{\mathrm{f}}, \sigma^{\gamma}, \sigma^{\mathrm{a}})$ to have either, but not both, of σ^{f} and σ^{γ} empty. (We will use this in small illustrative examples to follow.)

Definition 11 *Let $(\sigma^{\mathrm{f}}, \sigma^{\gamma}, \sigma^{\mathrm{a}})$ be an augmented action signature. s is an interpretation of the augmented action signature $(\sigma^{\mathrm{f}}, \sigma^{\gamma}, \sigma^{\mathrm{a}})$ when s is an interpretation of $\sigma^{\mathrm{f}} \cup \sigma^{\gamma}$ that agrees on the values of all σ^{a}-equivalent pairs of* $\mathsf{counts\_as}$ *atoms in σ^{γ}. We will write $\mathrm{I}(\sigma^{\mathrm{f}} \cup \sigma^{\gamma}; \sigma^{\mathrm{a}})$ for the set of all such interpretations. For convenience later, we write*

$$\begin{aligned} \mathit{Equiv}(\sigma^{\gamma}, \sigma^{\mathrm{a}}) =_{\mathrm{def}} \{ & \alpha\,\mathsf{counts\_as}\,\beta \leftrightarrow \alpha'\,\mathsf{counts\_as}\,\beta' \mid \\ & \alpha\,\mathsf{counts\_as}\,\beta \text{ and } \alpha'\,\mathsf{counts\_as}\,\beta' \text{ are distinct} \\ & \text{constants of } \sigma^{\gamma} \text{ such that } \alpha \equiv_{\sigma^{\mathrm{a}}} \alpha' \text{ and } \beta \equiv_{\sigma^{\mathrm{a}}} \beta' \} \end{aligned}$$

s is then an interpretation of $(\sigma^{\mathrm{f}}, \sigma^{\gamma}, \sigma^{\mathrm{a}})$ when s is an interpretation of $\sigma^{\mathrm{f}} \cup \sigma^{\gamma}$ such that $s \models \mathit{Equiv}(\sigma^{\gamma}, \sigma^{\mathrm{a}})$.

Here is the special case of interest.

Definition 12 (Augmented transition system (ATS)) *An* augmented transition system (ATS) *of signature $(\sigma^{\mathrm{f}}, \sigma^{\gamma}, \sigma^{\mathrm{a}})$ is a structure of the form*

$$\langle \sigma^{\mathrm{f}}, \sigma^{\gamma}, S, \mathrm{I}(\sigma^{\mathrm{a}}), R, C^{+}_{\mathcal{I}} \rangle$$

where

- *S is the set of augmented states, $S \subseteq \mathrm{I}(\sigma^{\mathrm{f}} \cup \sigma^{\gamma}; \sigma^{\mathrm{a}})$, i.e., every s in S is an interpretation of $\sigma^{\mathrm{f}} \cup \sigma^{\gamma}$ that agrees on the values of all σ^{a}-equivalent pairs of* $\mathsf{counts\_as}$ *atoms in σ^{γ}, $s \models \mathit{Equiv}(\sigma^{\gamma}, \sigma^{\mathrm{a}})$,*
- *$\mathrm{I}(\sigma^{\mathrm{a}})$ is the set of transition labels/events,*
- *R is the set of labelled transitions, $R \subseteq S \times \mathrm{I}(\sigma^{\mathrm{a}}) \times S$,*
- *$C^{+}_{\mathcal{I}}$ is the dynamic 'counts as' relation, $C^{+}_{\mathcal{I}} \subseteq S \times \wp(\mathrm{I}(\sigma^{\mathrm{a}})) \times \wp(\mathrm{I}(\sigma^{\mathrm{a}}))$, defined as*

$$C^{+}_{\mathcal{I}} =_{\mathrm{def}} \{\, (s, \|\alpha\|, \|\beta\|) \mid s \in S,\ s \models \alpha\,\mathsf{counts\_as}\,\beta \,\}$$

- *R satisfies the following condition, for all states $s \in S$ and all transition types $X, Y \subseteq \mathrm{I}(\sigma^{\mathrm{a}})$:*

$$\text{if } C^{+}_{\mathcal{I}}(s, X, Y) \text{ then for any } (s, \varepsilon, s') \in R, \text{ if } \varepsilon \in X \text{ then } \varepsilon \in Y \qquad (\mathrm{C}^{+}_{\varepsilon})$$

There is clearly some redundancy in this structure, because the $C_{\mathcal{I}}^{+}$ component is already determined by the augmented states, but keeping $C_{\mathcal{I}}^{+}$ explicit is convenient for expressing constraints on an ATS, and for ease of comparison with the static ATS structure of the previous section. The condition ($\mathrm{C}_{\varepsilon}^{+}$) is equivalently stated (see below) as:

$$\text{if } (s, \varepsilon, s') \in R \text{ then, for any } s \models \alpha \ \mathsf{counts\_as} \ \beta, \text{ if } \varepsilon \in \|\alpha\| \text{ then } \varepsilon \in \|\beta\|$$

or equivalently again

$$\text{if } (s, \varepsilon, s') \in R \text{ then, for any } s \models \alpha \ \mathsf{counts\_as} \ \beta, \text{ we have } \varepsilon \models \alpha \rightarrow \beta$$

Now, by construction, we have $(s, \|\alpha\|, \|\beta\|) \in C_{\mathcal{I}}$ iff $s \models \alpha \ \mathsf{counts\_as} \ \beta$, and so evaluation of $\alpha \ \mathsf{counts\_as}_{\mathcal{I}} \ \beta$ expressions in a query language evaluated on the augmented transition system can be reduced to evaluation of $\alpha \ \mathsf{counts\_as} \ \beta$ atoms on *states*. This is a very desirable feature.

Note again that we have chosen not to eliminate the possibility of unsupported institutional actions in the ATS but leave them to be eliminated as desired when action descriptions are formulated. We return to this point in Section 8.8.

Remark One can see that a static ATS $\langle \sigma^{\mathrm{f}}, \sigma^{\gamma}, S_{\mathrm{f}}, \mathbf{A}, R_{\mathrm{f}}, C_{\mathcal{I}} \rangle$ is the special case of an ATS $\langle \sigma^{\mathrm{f}}, \sigma^{\gamma}, S, \mathbf{A}, R, C_{\mathcal{I}}^{+} \rangle$ in which there are no states $s_{\mathrm{f}} \cup \gamma$ and $s_{\mathrm{f}} \cup \gamma'$ in S with $\gamma \neq \gamma'$, and where S_{f} is the set of states $\{s_{\mathrm{f}} \mid s_{\mathrm{f}} \cup \gamma \in S\}$, $R_{\mathrm{f}} = \{(s_{\mathrm{f}}, \varepsilon, s'_{\mathrm{f}}) \mid (s_{\mathrm{f}} \cup \gamma, \varepsilon, s'_{\mathrm{f}} \cup \gamma') \in R\}$, and $C_{\mathcal{I}} = \{(s_{\mathrm{f}}, X, Y) \mid (s_{\mathrm{f}} \cup \gamma, X, Y) \in C_{\mathcal{I}}^{+}\}$. We note that a dynamic ATS can be approximated by a static ATS, with no real loss of expressive power. For instead of allowing actions to change the values of counts_as constants, we can always define counts_as in terms of ordinary fluent constants, possibly with the introduction of new fluent constants. We then specify how actions affect these (new) fluent constants rather than the counts_as constants directly. Since the values of counts_as constants are then determined completely by the values of ordinary fluent constants in each state, a static 'counts as' relation is sufficient. It may seem therefore that there is no real benefit in adopting the more complicated dynamically augmented structure. However, there is also no real benefit in restricting attention to the simpler static version. As an implementation issue, the causal theories into which $(\mathcal{C}+)^{+}$ action descriptions are translated turn out to be essentially the same whether we choose the static or the dynamic version. Moreover, even in the static version, the 'counts as' relation $C_{\mathcal{I}}$ has to be represented somehow. If we choose to represent it by a set of counts_as atoms, which seems natural, then we end up with augmented states again, exactly as in the dynamic version but now without the flexibility to manipulate counts_as constants directly. We save nothing, at the price of having to impose a restriction on the $(\mathcal{C}+)^{+}$ language to prohibit formulas containing counts_as constants from appearing in the heads of fluent dynamic laws. We might as well support the dynamic notion, at no extra cost, and with the additional flexibility.

8.3 The language $(\mathcal{C}+)^+$

We want to be able to include 'counts as' expressions in both the heads and bodies of static and dynamic laws, and to write, for example, laws of the form α' causes $(\alpha$ counts_as $\beta)$ if G. Syntactically, expressions of the form α counts_as β will behave just like (Boolean) fluent constants. In any given application the set of action formulas α and β that can appear in a counts_as constant may be further restricted when the 'counts as' signature σ^γ is specified. In particular it is often sufficient to restrict α counts_as β constants to the form where both α and β are action *atoms*, or more generally conjunctions of action atoms, from some designated set. The general development does not depend on any such restriction however, so we will not adopt it yet, but defer further discussion until we consider the translation of $(\mathcal{C}+)^+$ action descriptions to causal theories. Moreover, to provide flexibility in formulating action descriptions, it is convenient to retain the distinction between *simple* and *statically determined* constants in the 'counts as' signature σ^γ.

Syntax The syntax of static and dynamic laws of $(\mathcal{C}+)^+$ is exactly as in the language $\mathcal{C}+$, including the restrictions that apply to statically determined constants, and including the abbreviations inertial, causes, etc., described earlier. Syntactically, an action description of $(\mathcal{C}+)^+$ of signature $(\sigma^{\mathrm{f}}, \sigma^\gamma, \sigma^{\mathrm{a}})$ is also an action description of $\mathcal{C}+$ of signature $(\sigma^{\mathrm{f}} \cup \sigma^\gamma, \sigma^{\mathrm{a}})$. Semantically, counts_as constants are given a special status in $(\mathcal{C}+)^+$ that they do not have in $\mathcal{C}+$ where they are just regular fluent constants.

To avoid (unintended) unsupported institutional actions, any action constant in the formula β of an α counts_as β atom is not (automatically) exogenous, though it may be deliberately declared as exogenous when the action description is formulated. (An example follows presently.)

Semantics An action description of $(\mathcal{C}+)^+$ defines an augmented transition system: the counts_as constants are given special treatment in that they define its 'counts as' relation, as follows.

Definition 13 *Let D be an action description of $(\mathcal{C}+)^+$ of signature $(\sigma^{\mathrm{f}}, \sigma^\gamma, \sigma^{\mathrm{a}})$. The ATS defined by D is*

$$\langle \sigma^{\mathrm{f}}, \sigma^\gamma, S, \mathrm{I}(\sigma^{\mathrm{a}}), R, C_D^+ \rangle$$

where:

- *$s \in S$ is a state of the ATS defined by D (in short: a $(\mathcal{C}+)^+$-defined state of D) iff s is the only interpretation of $\sigma^{\mathrm{f}} \cup \sigma^\gamma$ that agrees on the values of all σ^{a}-equivalent pairs of* counts_as *atoms in σ^γ and satisfies $T_{static}(s) \cup Simple(s)$, i.e., the only interpretation of $\sigma^{\mathrm{f}} \cup \sigma^\gamma$ such that*[3]

$$s \mid\!\!\models T_{static}(s) \cup Simple(s) \cup Equiv(\sigma^\gamma, \sigma^{\mathrm{a}})$$

[3]It is easy to check that s is the only interpretation of $\sigma^{\mathrm{f}} \cup \sigma^\gamma$ that agrees on the values of all σ^{a}-equivalent pairs of counts_as atoms in σ^γ and satisfies a set of formulas X iff $s \mid\!\!\models X \cup Equiv(\sigma^\gamma, \sigma^{\mathrm{a}})$.

- $(s, \varepsilon, s') \in R$ *is a transition of the ATS defined by* D *(in short: a* $(\mathcal{C}+)^+$*-defined transition of* D*) iff* s *and* s' *are interpretations of* $\sigma^{\mathrm{f}} \cup \sigma^{\gamma}$ *and* ε *is an interpretation of* σ^{a} *such that*
  - $s \not\models T_{static}(s) \cup Simple(s) \cup Equiv(\sigma^{\gamma}, \sigma^{\mathrm{a}})$
  - $s' \not\models T_{static}(s') \cup E(s, \varepsilon, s') \cup Equiv(\sigma^{\gamma}, \sigma^{\mathrm{a}})$
  - $\varepsilon \not\models A(\varepsilon, s) \cup Exog(\varepsilon) \cup \{\alpha \rightarrow \beta \mid s \models \alpha \,\mathsf{counts\_as}\, \beta\}$
- *The 'counts as' relation* C_D^+ *of the ATS defined by* D *(in short: the 'counts as' relation defined by* D*) is*

$$C_D^+ = \{(s, \|\alpha\|, \|\beta\|) \mid s \models \alpha \,\mathsf{counts\_as}\, \beta\}$$

Compare in particular the conditions $\varepsilon \not\models A(\varepsilon, s) \cup Exog(\varepsilon) \cup \{\alpha \rightarrow \beta \mid s \models \alpha \mathsf{counts\_as} \beta\}$ required for (s, ε, s') to be a transition of the $(\mathcal{C}+)^+$ action description D with the weaker conditions $\varepsilon \not\models A(\varepsilon, s) \cup Exog(\varepsilon)$ required for (s, ε, s') to be a transition of the $\mathcal{C}+$ action description D.

It remains to check that the transition system defined by a $(\mathcal{C}+)^+$ action description D is indeed an ATS of the correct form. We need to check that the transitions $(\mathcal{C}+)^+$-defined by D satisfy the condition $(\mathrm{C}_{\varepsilon}^+)$, and that when (s, ε, s') is a transition $(\mathcal{C}+)^+$-defined by D then s' is an augmented state of D.

Proposition 14 *Let* D *be an action description of* $(\mathcal{C}+)^+$*. The transition system* $\langle \sigma^{\mathrm{f}}, \sigma^{\gamma}, S, \mathrm{I}(\sigma^{\mathrm{a}}), R, C_D^+ \rangle$ *defined by* D *(i) satisfies the condition* $(\mathrm{C}_{\varepsilon}^+)$ *for an ATS, and (ii) has the property that when* $(s, \varepsilon, s') \in R$ *then* $s' \in S$*.*

Proof. For (i): suppose $(s, X, Y) \in C_D^+$. Then $X = \|\alpha\|$ and $Y = \|\beta\|$ for some action formulas α and β such that $s \models \alpha \,\mathsf{counts\_as}\, \beta$. Suppose $(s, \varepsilon, s') \in R$. Then by Definition 13, we have $\varepsilon \models (\alpha \rightarrow \beta)$, i.e., $\varepsilon \in \|\alpha\|$ implies $\varepsilon \in \|\beta\|$, and so $\varepsilon \in X$ implies $\varepsilon \in Y$, as required.

For (ii), we need to show that if s' is the only model of $T_{static}(s') \cup E(s, \varepsilon, s') \cup Equiv(\sigma^{\gamma}, \sigma^{\mathrm{a}})$ then s' is the only model of $T_{static}(s') \cup Simple(s') \cup Equiv(\sigma^{\gamma}, \sigma^{\mathrm{a}})$. This follows from the observation that $E(s, \varepsilon, s')$ contains no statically determined constants, by exactly the same line of reasoning as in the proof of Proposition 4 for labelled transition systems. □

Example (waving a flag) Consider the $(\mathcal{C}+)^+$ action description D with Boolean action constants *wave* and *start*, where *wave* (representing the waving of a flag, say) is exogenous and *start* (representing the institutional action of starting a race, say) is not. σ^{γ} contains the simple (Boolean) constant *wave* counts_as *start*. Suppose D contains just the static laws

$$wave \,\mathsf{counts\_as}\, start \;\mathsf{if}\; wave \,\mathsf{counts\_as}\, start$$
$$\neg(wave \,\mathsf{counts\_as}\, start) \;\mathsf{if}\; \neg(wave \,\mathsf{counts\_as}\, start)$$

(i.e., exogenous *wave* counts_as *start*) and the action dynamic law

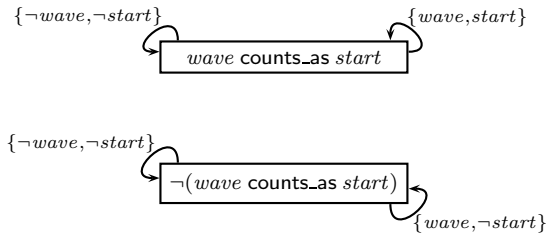
$$\neg start \;\mathsf{if}\; \neg start \quad (\text{i.e., } \mathsf{default}\ \neg start)$$

There are no simple fluent constants in σ^{f}, no trivial constants in either σ^{f} or σ^{γ}, no pairs of distinct σ^{a}-equivalent counts_as constants in σ^{γ}, and there are

only two static laws, so it is easy to confirm that there are precisely two states in the ATS defined by D:

$$\{wave \text{ counts\_as } start\} \quad \text{and} \quad \{\neg(wave \text{ counts\_as } start)\}.$$

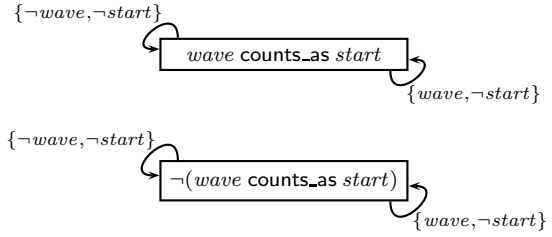
There are no trivial constants in σ^{a} and only one action dynamic law, and so the transitions in the ATS defined by D can be easily determined, as shown in the diagram below.



$(\mathcal{C}+)^+$: default $\neg start$

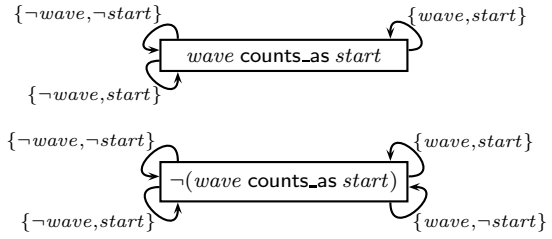
When *wave* counts_as *start* holds, all transitions of type *wave* are also transitions of type *start*. Notice that default $\neg start$ is strong enough (in this example) to eliminate unsupported institutional actions $\{\neg wave, start\}$ even when *wave* counts_as *start* does not hold.

For comparison, here is the transition system obtained when the same action description is viewed as an action description of $\mathcal{C}+$.



$\mathcal{C}+$: default $\neg start$

We can admit unsupported institutional actions in the $(\mathcal{C}+)^+$-defined ATS, if we want them, by making *start* exogenous, effectively adding the action dynamic law *start* if *start* to default $\neg start$. We then get



$(\mathcal{C}+)^+$: exogenous *start*

This is the transition system that would be obtained from the $\mathcal{C}+$ action description, except that the $\mathcal{C}+$-defined system also contains a $\{wave, \neg start\}$ transition from $\{wave \text{ counts\_as } start\}$ to itself.

Note that the elimination of unsupported institutional actions in this example does not depend on the fact that there is only one means of starting a race. Suppose we extend the example with another action constant *shoot* (for shooting a gun, say) and another constant *shoot* counts_as *start* in the action signature. We add the static laws exogenous *shoot* counts_as *start*. There are four states in the ATS so defined. The state $\{wave \text{ counts\_as } start,\ shoot \text{ counts\_as } start\}$ has (reflexive) transitions $\{wave, shoot, start\}$, $\{wave, \neg shoot, start\}$, $\{\neg wave, shoot, start\}$, and $\{\neg wave, \neg shoot, \neg start\}$ but no transition $\{\neg wave, \neg shoot, start\}$ where there is an unsupported institutional action. The state $\{wave \text{ counts\_as } start, \neg(shoot \text{ counts\_as } start)\}$ has (reflexive) transitions $\{wave, shoot, start\}$, $\{wave, \neg shoot, start\}$, and $\{\neg wave, \neg shoot, \neg start\}$, but no transition $\{\neg wave, shoot, start\}$ and no transition $\{\neg wave, \neg shoot, start\}$ with an unsupported institutional action. Similarly for $\{\neg(wave \text{ counts\_as } start),\ shoot \text{ counts\_as } start\}$. The state $\{\neg(wave \text{ counts\_as } start),\ \neg(shoot \text{ counts\_as } start)\}$ has (reflexive) transitions $\{wave, shoot, \neg start\}$, $\{wave, \neg shoot, \neg start\}$, $\{\neg wave, shoot, \neg start\}$, and $\{\neg wave, \neg shoot, \neg start\}$ but no transitions of type *start*, and in particular no transition $\{\neg wave, \neg shoot, start\}$ with an unsupported institutional action.

We have eliminated unsupported institutional actions in this example by means of the action dynamic law default $\neg start$. This was chosen to emphasise the difference between $(\mathcal{C}+)^+$ and $\mathcal{C}+$ action descriptions. It is simpler and clearer to eliminate institutional actions by means of a law of the form

$$\text{nonexecutable } start \wedge \neg wave \wedge \neg shoot$$

This is the style we will adopt in subsequent examples.

It is now possible to investigate the relationships that exist in general between the states and transitions of a $(\mathcal{C}+)^+$ action description of signature $(\sigma^{\mathrm{f}}, \sigma^{\gamma}, \sigma^{\mathrm{a}})$ and those of the $\mathcal{C}+$ action description of signature $(\sigma^{\mathrm{f}} \cup \sigma^{\gamma}, \sigma^{\mathrm{a}})$. These relationships are somewhat involved however and we omit the details here.

8.4 Translation to $\mathcal{C}+$ and causal theories

Now: given an action description D of $(\mathcal{C}+)^+$ we construct a causal theory ${\Gamma^+}_m^D$ such that models of ${\Gamma^+}_m^D$ correspond one-to-one with the paths of length m of the ATS defined by D. We do not need to worry about representing the 'counts as' relation C_D^+ in models of ${\Gamma^+}_m^D$ because counts_as atoms in states do that for us already.

In fact, we can define a translation from a $(\mathcal{C}+)^+$ action description of signature $(\sigma^{\mathrm{f}}, \sigma^{\gamma}, \sigma^{\mathrm{a}})$ to a $\mathcal{C}+$ action description $D \cup \delta$ of signature $(\sigma^{\mathrm{f}} \cup \sigma^{\gamma}, \sigma^{\mathrm{a}})$ where the additional $\mathcal{C}+$ laws δ capture the special treatment given to counts_as constants in $(\mathcal{C}+)^+$. Models of the causal theory $\Gamma_m^{D\cup\delta}$ obtained from the $\mathcal{C}+$ action description $D \cup \delta$ correspond one-to-one with the paths of length m of the ATS defined by D; in other words, ${\Gamma^+}_m^D =_{\mathrm{def}} \Gamma_m^{D\cup\delta}$. Moreover (subject to some restrictions identified below) when D is a definite action description of

$(\mathcal{C}+)^+$, the additional laws δ are all definite laws of $\mathcal{C}+$, and so we obtain the very desirable property that definite action descriptions of $(\mathcal{C}+)^+$ translate to definite action descriptions of $\mathcal{C}+$. This means that we can employ the method of literal completion, and the Causal Calculator CCALC, to perform computations on definite $(\mathcal{C}+)^+$ action descriptions.

The additional causal laws δ required to translate a $(\mathcal{C}+)^+$ action description D to the equivalent $\mathcal{C}+$ action description consist of two (disjoint) sets, δ_D and δ_{σ^γ}. The basic idea is that δ_D contains a $\mathcal{C}+$ action dynamic law

$$\beta \;\; \textsf{if} \;\; \alpha \wedge \alpha \,\textsf{counts\_as}\, \beta$$

for every constant $\alpha\textsf{counts\_as}\beta$ in σ^γ. However, in the case where β is a conjunction of (not necessarily atomic) action formulas it is much more convenient to include a separate action dynamic law for each conjunct of β, dealing separately with the special cases where the conjunct is $\bot$ or $\top$. More precisely: every formula β in a constant $\alpha\,\textsf{counts\_as}\,\beta$ of σ^γ is a conjunction $\beta_1 \wedge \cdots \wedge \beta_m$ $(m \geq 1)$ of (not necessarily atomic) formulas β_i. δ_D is the set of $\mathcal{C}+$ laws obtained as follows. For every constant $\alpha\,\textsf{counts\_as}\,\beta$ in σ^γ such that $\bot$ is not a conjunct of β, δ_D includes a $\mathcal{C}+$ action dynamic law of the form

$$\beta_i \;\; \textsf{if} \;\; \alpha \wedge \alpha \,\textsf{counts\_as}\, \beta \tag{9}$$

for every conjunct $\beta_i \neq \top$ of β; and for every constant $\alpha\,\textsf{counts\_as}\,\beta$ in σ^γ such that $\bot$ is a conjunct of β, δ_D includes a $\mathcal{C}+$ fluent dynamic law of the form

$$\bot \;\textsf{if}\; \top \;\textsf{after}\; \alpha \wedge \alpha \,\textsf{counts\_as}\, \beta \tag{10}$$

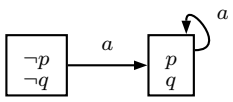
(i.e., $\textsf{nonexecutable}\ \alpha\ \textsf{if}\ \alpha\,\textsf{counts\_as}\,\beta$ in abbreviated form). The separate form (10) is required in the latter case because the expression $\bot\ \textsf{if}\ \alpha \wedge \alpha\,\textsf{counts\_as}\,\beta$ is not a well-formed action dynamic law of $\mathcal{C}+$ ($\bot$ is a 0-ary connective and not an action formula of σ^{a}). And note that although there is no constant of the form $\alpha\,\textsf{counts\_as}\,\top$ in σ^γ ($\top$ is a 0-ary connective) there could be a conjunct $\top$ in the formula β of a constant $\alpha\,\textsf{counts\_as}\,\beta$. No $\mathcal{C}+$ law is included in δ_D in this case. ($\top\ \textsf{if}\ \top\ \textsf{after}\ \alpha\,\textsf{counts\_as}\,\beta$ is well-formed but it has no effect on the models of a $\mathcal{C}+$ action description and so is omitted from δ_D.)

The other component δ_{σ^γ} of δ is the set of $\mathcal{C}+$ static laws of the form

$$(\alpha \,\textsf{counts\_as}\, \beta \leftrightarrow \alpha' \,\textsf{counts\_as}\, \beta') \;\; \textsf{if} \;\; \top \tag{11}$$

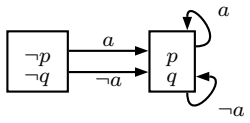
for every pair of distinct constants $\alpha\textsf{counts\_as}\beta$ and $\alpha'\textsf{counts\_as}\beta'$ in σ^γ such that $\alpha \equiv_{\sigma^{\mathrm{a}}} \alpha'$ and $\beta \equiv_{\sigma^{\mathrm{a}}} \beta'$. These are required to ensure that interpretations/states agree on the values of σ^{a}-equivalent $\textsf{counts\_as}$ constants.

It may seem that the causal laws δ_{σ^γ} are unnecessarily cumbersome to get the desired effect. But consider: suppose we want simple Boolean constants p and q to have the same values in all states. (We choose p and q here to make the point rather than $\textsf{counts\_as}$ atoms to reduce writing.) Suppose, for illustration, that we have the single fluent dynamic law $a\ \textsf{causes}\ p$. Adding the static law $(p \leftrightarrow q)\ \textsf{if}\ \top$ gives the following set of states and transitions

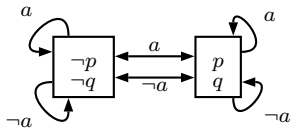


which is also what is required by the semantics.

If in place of $(p \leftrightarrow q)$ if $\top$ we try the pair of static laws $\bot$ if $p \wedge \neg q$ and $\bot$ if $q \wedge \neg p$ we get the right states but we get no transitions. (There is nothing to 'cause' q in the state after a.) If we try instead the pair of static laws p if q and q if p we again get the right states but now *too many* transitions. We have the implied laws p if p and q if q, and we get the following states and transitions:



If we add two further laws $\neg p$ if $\neg q$ and $\neg q$ if $\neg p$ then p and q both become (implied) exogenous, and we get even more transitions:



The form of causal laws δ_D in (9) and (10) is chosen so that when D is a definite action description of $(\mathcal{C}+)^+$ (defined below), and $Equiv(\sigma^\gamma, \sigma^{\mathrm{a}})$ and δ_{σ^γ} are empty, then $D \cup \delta_D = D \cup \delta_D \cup \delta_{\sigma^\gamma}$ is a definite action description of $\mathcal{C}+$. There will be no causal laws of form (10) in δ_D in this case. However, the soundness of the translation from $(\mathcal{C}+)^+$ to $\mathcal{C}+$ for the general case does not depend on any assumptions about the form of β in α counts_as β constants.

Theorem 15 *Let D be a $(\mathcal{C}+)^+$ action description of signature $(\sigma^{\mathrm{f}}, \sigma^\gamma, \sigma^{\mathrm{a}})$. Let δ_D be the set of $\mathcal{C}+$ action dynamic laws*

$$\beta_i \quad \text{if} \quad \alpha \ \wedge \ \alpha \,\mathsf{counts\_as}\, \beta_1 \wedge \cdots \wedge \beta_i \wedge \cdots \wedge \beta_m$$

for every constant α counts_as $\beta_1 \wedge \cdots \wedge \beta_i \wedge \cdots \wedge \beta_m$ $(m \geq 1)$ in σ^γ for which $\beta_k \neq \bot$ for any $1 \leq k \leq m$ and every β_i $(1 \leq i \leq m)$ such that $\beta_i \neq \top$, together with $\mathcal{C}+$ fluent dynamic laws

$$\bot \ \text{if} \ \top \quad \text{after} \ \alpha \ \wedge \ \alpha \,\mathsf{counts\_as}\, \beta$$

for every constant α counts_as β in σ^γ for which $\bot$ is a conjunct of β. Let δ_{σ^γ} be the set of $\mathcal{C}+$ static laws

$$(\alpha \,\mathsf{counts\_as}\, \beta \leftrightarrow \alpha' \,\mathsf{counts\_as}\, \beta') \quad \text{if} \quad \top$$

for every pair of distinct constants α counts_as β and α' counts_as β' in σ^γ such that $\alpha \equiv_{\sigma^{\mathrm{a}}} \alpha'$ and $\beta \equiv_{\sigma^{\mathrm{a}}} \beta'$.

s is a state of the ATS defined by D iff s is a state of the labelled transition system defined by the $\mathcal{C}+$ action description $D \cup \delta_D \cup \delta_{\sigma^\gamma}$ of signature $(\sigma^{\mathrm{f}} \cup \sigma^\gamma, \sigma^{\mathrm{a}})$.

(s, ε, s') is a transition of the ATS defined by D iff (s, ε, s') is a transition of the labelled transition system defined by the $\mathcal{C}+$ action description $D \cup \delta_D \cup \delta_{\sigma^\gamma}$ of signature $(\sigma^{\mathrm{f}} \cup \sigma^\gamma, \sigma^{\mathrm{a}})$.

Proof. The expressions $T_{static}(s)$, $E(s, \varepsilon, s')$, $A(\varepsilon, s)$ refer here to the causal laws of the $(\mathcal{C}+)^+$ action description D.

States: The conditions for $s \in S$ to be a $(\mathcal{C}+)^+$-defined augmented state of D are:

$$s \mathrel{!\models} T_{static}(s) \cup \mathit{Simple}(s) \cup \mathit{Equiv}(\sigma^\gamma, \sigma^\mathrm{a})$$

Compare the conditions for s to be a state of the $\mathcal{C}+$ action description $D \cup \delta_D \cup \delta_{\sigma^\gamma}$:

$$s \mathrel{!\models} \big(T_{static}(s) \cup \mathit{Equiv}(\sigma^\gamma, \sigma^\mathrm{a})\big) \cup \mathit{Simple}(s)$$

These are clearly equivalent.

Transitions: To prove the result for transitions we show that the conditions

$$\begin{aligned} s' &\mathrel{!\models} T_{static}(s') \cup E(s, \varepsilon, s') \cup \mathit{Equiv}(\sigma^\gamma, \sigma^\mathrm{a}) \\ \varepsilon &\mathrel{!\models} A(\varepsilon, s) \cup \mathit{Exog}(\varepsilon) \cup \{\alpha \rightarrow \beta \mid s \models \alpha \ \mathsf{counts\_as}\ \beta\} \end{aligned}$$

are equivalent to the conditions

$$\begin{aligned} s' &\mathrel{!\models} \big(T_{static}(s') \cup \mathit{Equiv}(\sigma^\gamma, \sigma^\mathrm{a})\big) \cup \big(E(s, \varepsilon, s') \cup E_{\delta_D}(s, \varepsilon, s')\big) \\ \varepsilon &\mathrel{!\models} \big(A(\varepsilon, s) \cup A_{\delta_D}(\varepsilon, s)\big) \cup \mathit{Exog}(\varepsilon) \end{aligned}$$

where $E_{\delta_D}(s, \varepsilon, s')$ and $A_{\delta_D}(\varepsilon, s')$ denote respectively the heads of the fluent dynamic and action dynamic laws in δ_D whose bodies are satisfied by the transition (s, ε, s'), i.e.,

$$\begin{aligned} E_{\delta_D}(s, \varepsilon, s') &= \{\bot \mid \alpha \ \mathsf{counts\_as}\ \beta \text{ in } \sigma^\gamma,\ \bot \text{ is a conjunct of } \beta, \\ &\qquad\qquad s' \models \top,\ s \cup \varepsilon \models \alpha \ \wedge\ \alpha \ \mathsf{counts\_as}\ \beta\} \\ &= \{\bot \mid s \models \alpha \ \mathsf{counts\_as}\ \beta,\ \bot \text{ is a conjunct of } \beta,\ \varepsilon \models \alpha\} \end{aligned}$$

and

$$\begin{aligned} A_{\delta_D}(\varepsilon, s) &= \{\beta_i \mid \alpha \ \mathsf{counts\_as}\ \beta \text{ in } \sigma^\gamma,\ \bot \text{ is not a conjunct of } \beta, \\ &\qquad\qquad \beta_i \text{ is a conjunct of } \beta,\ \beta_i \neq \top,\ s \cup \varepsilon \models \alpha \ \wedge\ \alpha \ \mathsf{counts\_as}\ \beta\} \\ &= \{\beta_i \mid s \models \alpha \ \mathsf{counts\_as}\ \beta,\ \bot \text{ is not a conjunct of } \beta, \\ &\qquad\qquad \beta_i \text{ is a conjunct of } \beta,\ \beta_i \neq \top,\ \varepsilon \models \alpha\} \end{aligned}$$

The conditions on s' and ε above are equivalently expressed (because s' and ε are disjoint) as

$$\begin{aligned} s' \cup \varepsilon \mathrel{!\models} T_{static}(s') \cup \mathit{Equiv}(\sigma^\gamma, \sigma^\mathrm{a}) \cup E(s, \varepsilon, s') \cup E_{\delta_D}(s, \varepsilon, s') \cup \\ A(\varepsilon, s) \cup A_{\delta_D}(\varepsilon, s) \cup \mathit{Exog}(\varepsilon) \end{aligned}$$

which in turn can be expressed (because s' and ε are disjoint) as

$$\begin{aligned} s' &\mathrel{!\models} T_{static}(s') \cup E(s, \varepsilon, s') \cup \mathit{Equiv}(\sigma^\gamma, \sigma^\mathrm{a}) \\ \varepsilon &\mathrel{!\models} A(\varepsilon, s) \cup \mathit{Exog}(\varepsilon) \cup A_{\delta_D}(\varepsilon, s) \cup E_{\delta_D}(s, \varepsilon, s') \end{aligned}$$

Now

$$\begin{aligned} \varepsilon \models A_{\delta_D}(\varepsilon, s) \text{ iff } \varepsilon &\models \{\beta_i \mid s \models \alpha \ \mathsf{counts\_as}\ \beta,\ \bot \text{ is not a conjunct of } \beta, \\ &\qquad\qquad \beta_i \text{ is a conjunct of } \beta,\ \beta_i \neq \top,\ \varepsilon \models \alpha\} \\ \text{iff } \varepsilon &\models \{\beta_i \mid s \models \alpha \ \mathsf{counts\_as}\ \beta,\ \bot \text{ is not a conjunct of } \beta, \\ &\qquad\qquad \beta_i \text{ is a conjunct of } \beta,\ \varepsilon \models \alpha\} \end{aligned}$$

and

$$\begin{aligned}\varepsilon \models E_{\delta_D}(\varepsilon, s) \text{ iff } & \varepsilon \models \{\bot \mid s \models \alpha \,\mathsf{counts\_as}\, \beta,\ \bot \text{ is a conjunct of } \beta,\ \varepsilon \models \alpha\} \\ \text{iff } & \varepsilon \models \{\beta_i \mid s \models \alpha \,\mathsf{counts\_as}\, \beta,\ \bot \text{ is a conjunct of } \beta, \\ & \qquad \beta_i \text{ is a conjunct of } \beta,\ \varepsilon \models \alpha\}\end{aligned}$$

And so

$$\begin{aligned}\varepsilon \models & A_{\delta_D}(\varepsilon, s) \cup E_{\delta_D}(\varepsilon, s) \\ & \text{iff } \varepsilon \models \{\beta_i \mid s \models \alpha \,\mathsf{counts\_as}\, \beta,\ \beta_i \text{ is a conjunct of } \beta,\ \varepsilon \models \alpha\} \\ & \text{iff } \varepsilon \models \{\beta \mid s \models \alpha \,\mathsf{counts\_as}\, \beta,\ \varepsilon \models \alpha\} \\ & \text{iff } \varepsilon \models \{\alpha \rightarrow \beta \mid s \models \alpha \,\mathsf{counts\_as}\, \beta\}\end{aligned}$$

which completes the proof. □

Corollary 16 *Paths of length m in the ATS defined by a $(\mathcal{C}+)^+$ action description D are in one-to-one correspondence with the models of the causal theory $\Gamma_m^{D \cup \delta_D \cup \delta_{\sigma^\gamma}}$.*

8.5 Reduced action descriptions

The causal laws δ_{σ^γ} required for translation to $\mathcal{C}+$ are a nuisance—for one thing, they mean that in general a $(\mathcal{C}+)^+$ action description cannot translate to a definite action description of $\mathcal{C}+$. They are included primarily for the sake of completeness. There are a number of straightforward methods by which their effect can be obtained in other ways, depending on the level of generality required.

One obvious method is as follows. We want to restrict attention to interpretations that agree on the values of constants $c_1, c_2, \ldots, c_n$. In the present context $c_1, c_2, \ldots, c_n$ are σ^{a}-equivalent $\mathsf{counts\_as}$ constants in σ^γ. Instead of representing such interpretations by the set of all atoms they satisfy, it is obviously sufficient to represent an interpretation by the value of a single representative member of $c_1, c_2, \ldots, c_n$. So: we partition the $\mathsf{counts\_as}$ constants in σ^γ into σ^{a}-equivalent classes and pick one representative member from each equivalence class. This representative member can be identified when the signature σ^γ is specified or it can be chosen arbitrarily. Now we replace every occurrence of a $\mathsf{counts\_as}$ constant in the action description by the representative member of the σ^{a}-equivalence class to which it belongs. This is straightforward to implement, and has the additional benefit that the representation of interpretations is more concise. It only remains to ensure that the evaluation of query languages on the ATS similary translates formulas before they are evaluated, which is also easy to arrange.

We will call a $(\mathcal{C}+)^+$ action description D of signature $(\sigma^{\mathrm{f}}, \sigma^\gamma, \sigma^{\mathrm{a}})$ *reduced* if there are no distinct constants $\alpha \,\mathsf{counts\_as}\, \beta$ and $\alpha' \,\mathsf{counts\_as}\, \beta'$ in σ^γ such that $\alpha \equiv_{\sigma^{\mathrm{a}}} \alpha'$ and $\beta \equiv_{\sigma^{\mathrm{a}}} \beta'$. Clearly every action description of $(\mathcal{C}+)^+$ can be translated to a reduced action description, for example by the process outlined above. For a reduced action description D the set $\mathit{Equiv}(\sigma^\gamma, \sigma^{\mathrm{a}})$ is empty and so the conditions for s to be a state of the ATS $(\mathcal{C}+)^+$-defined by D are simply

$s \not\models T_{static}(s) \cup Simple(s)$ (as for a $\mathcal{C}+$ action description), and the translation of the $(\mathcal{C}+)^+$ action description to $\mathcal{C}+$ can dispense with the static laws δ_{σ^γ} since these are also empty.

In what follows, we will not employ this general method but instead restrict attention to a special form. We will say that an action description of signature $(\sigma^{\mathrm{f}}, \sigma^\gamma, \sigma^{\mathrm{a}})$, is *atom-conjunctive* when every $\alpha\,\mathsf{counts\_as}\,\beta$ constant of σ^γ is such that the formulas α and β are both (non-empty) conjunctions of action *atoms*, written in some standard order and without repetitions. As an implementation issue, it is easy to ensure that the conjuncts in counts_as constants are rearranged into the standard order if necessary in a simple pre-processing stage. Clearly all atom-conjunctive action descriptions are reduced. (And as a minor consideration, the translation of $(\mathcal{C}+)^+$ to $\mathcal{C}+$ does not introduce any fluent dynamic laws of the 'nonexecutable' form (10).)

Why this particular form? Although in practice it will often be the case that both α and β in $\alpha\,\mathsf{counts\_as}\,\beta$ constants are *atoms* of σ^{a}, we want to be able to support the more general case where they are conjunctions of atoms, for the following main reason. An action of type a with attributes $x_1, x_2, \ldots, x_n$ can be represented in $\mathcal{C}+$ (and $(\mathcal{C}+)^+$) either as an action atom $a(x_1, x_2, \ldots, x_n) = v$ with n arguments or as a conjunction of $n+1$ action atoms $a = v \wedge a_1 = x_1 \wedge a_2 = x_2 \wedge \cdots \wedge a_n = x_n$ where $a_1, \ldots, \alpha_n$ are (usually non-exogenous) action constants which pick out the values of the attributes. This second style of representation has significant advantages, both in terms of flexibility or 'elaboration tolerance' (Giunchiglia et al., 2004, Sect. 5.6), and economy, in that the number of action constants required in the first style is exponential in the number of attributes n whereas in the conjunction-of-attributes style it is linear.

The atom-conjunctive form accommodates all the examples that we can expect to encounter in practice. Moreover, it is likely that the properties of the 'counts as' relation $C^+_{\mathcal{I}}$ of an ATS can be strengthened quite naturally, allowing all $\alpha\,\mathsf{counts\_as}\,\beta$ atoms, for arbitrary action formulas α, to be written equivalently in a normal form as a truth-functional compound of $\alpha'\mathsf{counts\_as}\beta$ atoms in which α' is a conjunction of action atoms. This requires only that $(\alpha_1 \vee \alpha_2)\,\mathsf{counts\_as}\,\beta$ comes out to be equivalent to the conjunction of $\alpha_1\mathsf{counts\_as}\beta$ and $\alpha_2\mathsf{counts\_as}\beta$, which seems non-controversial. However, this remains to be checked, and we leave the details to a future version.

8.6 Definite action descriptions

As in the case of $\mathcal{C}+$, the action descriptions of primary interest are those that are definite.

Definition 17 *A $(\mathcal{C}+)^+$ action description D of signature $(\sigma^{\mathrm{f}}, \sigma^\gamma, \sigma^{\mathrm{a}})$ is* definite *iff the $\mathcal{C}+$ action description D of signature $(\sigma^{\mathrm{f}} \cup \sigma^\gamma, \sigma^{\mathrm{a}})$ is definite, and for every constant $\alpha\,\mathsf{counts\_as}\,\beta$ in σ^γ, β is a (non-empty) conjunction of atoms of σ^{a}, written in some standard order and without repetitions.*

Note that this definition does not depend on the form of action formulas α in $\alpha\,\mathsf{counts\_as}\,\beta$ constants, so an action description can be definite without being atom-conjunctive (or even 'reduced'). However, as already explained, we are

primarily interested in the special category of definite action descriptions which are also atom-conjunctive.

For definite, atom-conjunctive action descriptions, the characterisation of states and transitions of the ATS they define can be simplified as usual. For reference, the relevant conditions are as follows.

Proposition 18 *Let D be a definite, atom-conjunctive action description of $(\mathcal{C}+)^+$ of signature $(\sigma^{\mathrm{f}}, \sigma^{\gamma}, \sigma^{\mathrm{a}})$. s is a state of D and (s, ε, s') is a transition of D iff s and s' are interpretations of $\sigma^{\mathrm{f}} \cup \sigma^{\gamma}$ and ε is an interpretation of σ^{a} such that:*

- $s = T_{static}(s) \cup \mathit{Simple}(s) \cup \mathit{Trivial}(\sigma^{\mathrm{f}} \cup \sigma^{\gamma})$
- $s' = T_{static}(s') \cup E(s, \varepsilon, s') \cup \mathit{Trivial}(\sigma^{\mathrm{f}} \cup \sigma^{\gamma})$
- $\varepsilon = A(\varepsilon, s) \cup \mathit{Exog}(\varepsilon) \cup \mathit{Trivial}(\sigma^{\mathrm{a}}) \cup$
  $\{\, \beta_i \mid s \models \alpha \,\mathsf{counts\_as}\, \beta_1 \wedge \cdots \wedge \beta_i \wedge \cdots \wedge \beta_n,\ \varepsilon \models \alpha \,\}$

Further, $D \cup \delta_D$ is a definite action description of $\mathcal{C}+$ when D is a definite action description of $(\mathcal{C}+)^+$, and so we have the following.

Proposition 19 *Let D be a definite, atom-conjunctive action description of $(\mathcal{C}+)^+$ of signature $(\sigma^{\mathrm{f}}, \sigma^{\gamma}, \sigma^{\mathrm{a}})$. Paths of length m of the ATS defined by D are in one-to-one correspondence with the (classical) models of $\mathit{comp}(\Gamma_m^{D \cup \delta_D})$. In particular, states of the ATS are encoded by the models of $\mathit{comp}(\Gamma_0^{D \cup \delta_D})$ and transitions by the models of $\mathit{comp}(\Gamma_1^{D \cup \delta_D})$.*

Example Consider the following example. It is rather artificial but it is difficult to construct a convincing example without making it too large and complicated. A more realistic example is presented in Section 8.7 below.

Signature: exogenous action constants *wave_flag* and *wave_hand* (Boolean) and *switch* with values *on* and *off*; non-exogenous action constants *start_race*, *signalX* (Boolean); simple fluent constants *statusX* (Boolean) and *light* with values *on* and *off*; simple (Boolean) constants *wave_flag* counts_as *start_race*, $\mathit{switch} = \mathit{on}$ counts_as *signalX*, *wave_hand* counts_as *signalX*.

The following is a definite, atom-conjunctive action description of $(\mathcal{C}+)^+$:

inertial *light*
inertial *wave_flag* counts_as *start_race*
inertial $\mathit{switch} = \mathit{on}$ counts_as *signalX*
inertial *wave_hand* counts_as *signalX*
inertial *statusX*

$\mathit{switch} = \mathit{on}$ causes $\mathit{light} = \mathit{on}$
$\mathit{switch} = \mathit{off}$ causes $\mathit{light} = \mathit{off}$

wave_flag counts_as *start_race* if $\top$ % fixed, for the sake of the example

start_race causes ($\mathit{switch} = \mathit{on}$ counts_as *signalX*)
start_race causes (*wave_hand* counts_as *signalX*)

signalX causes *statusX*

nonexecutable $\mathit{start\_race} \wedge \neg \mathit{wave\_flag}$
nonexecutable $\mathit{signalX} \wedge \neg \mathit{switch} = \mathit{on} \wedge \neg \mathit{wave\_hand}$

(We allow the possibility of waving the flag, waving a hand, and switching the light at the same time.)

The translation to $\mathcal{C}+$ adds the following action dynamic laws:

$$\begin{array}{l} \mathit{start\_race} \text{ if } \mathit{wave\_flag} \wedge (\mathit{wave\_flag} \text{ counts\_as } \mathit{start\_race}) \\ \mathit{signalX} \text{ if } \mathit{switch} = \mathit{on} \wedge (\mathit{switch} = \mathit{on} \text{ counts\_as } \mathit{signalX}) \\ \mathit{signalX} \text{ if } \mathit{wave\_hand} \wedge (\mathit{wave\_hand} \text{ counts\_as } \mathit{signalX}) \end{array}$$

In the ATS so defined, there are no transitions of type $\mathit{wave\_flag} \wedge \neg \mathit{start\_race}$: if a *wave_flag* action occurs then the transition must also be of type *start_race*. And since *start_race* 'causes' $\mathit{switch} = \mathit{on}$ counts_as *signalX*, after a *wave_flag* event, there are no transitions of type $\mathit{switch} = \mathit{on} \wedge \neg \mathit{signalX}$: after a *wave_flag* event, any transition of type $\mathit{switch} = \mathit{on}$ is also of type *signalX*; its effects are those it has by virtue of being a transition of type $\mathit{switch} = \mathit{on}$ ($\mathit{light} = \mathit{on}$) and those it has by virtue of being a transition of type *signalX* (*statusX* is true). Likewise for *wave_hand* transitions following a *wave_flag*. The two nonexecutable laws of the action description are included to eliminate unsupported institutional actions.

8.7 Example: an auction protocol

Consider a simple kind of auction in which agents make bids as follows. Any agent can *open* the bidding (when there is no current bid) at an amount N up to some designated maximum. Any agent can *raise* the current bid by N units, where again N must not exceed the designated maximum. Bids can be made by agents in any order, except that no agent may bid twice in succession. An agent may *revoke* its last bid, but only until the next (valid) bid is made. An agent may *withdraw* from the auction at any time, unless it has made the last (valid) bid. Once withdrawn, an agent may not resume bidding. (The 'winner' is the last agent left in the auction when all others have withdrawn, though for simplicity we will ignore this here.)

The terms 'may' and 'can' are to be understood not as permission, but as part of the definition of what counts as a valid bid. The representation should be able to express both attempted but possibly invalid bids, and valid bids.

We have trivial (and so 'rigid') fluent constants $\mathit{player}(X)$ for X ranging over the names of participants in the auction. There is also a trivial fluent atom $\mathit{max\_raise} = \mathit{Max}$ for some positive integer *Max*. The action constants are Boolean constants (X *signals* α) and (X:α) for α ranging over *open* (N), *raise* (N), *withdraw*, *revoke*, and N ranging over all integers between 1 and *Max*. The action constants (X *signals* α) are exogenous, and (X:α) are not. We could also have made (X *signals open*), (X *signals raise*), (X:*open*) and (X:*raise*) multi-valued constants with values in the range $1..M$ for some integer M but there is little to be gained from doing so in this example.

To represent the state of the auction we have (inertial) multi-valued fluent constants *current_bidder* and *current_bid*, and an inertial (Boolean) fluent constant *withdrawn*(X) for each player X. The possible values of *current_bid* are all integers in the range $1..M_{\max}$ for some suitably chosen integer $M_{\max}$.

We also employ the abbreviation

$$\mathsf{pow}(X, \alpha) =_{\text{def}} (X \textit{ signals } \alpha) \text{ counts\_as } (X{:}\alpha)$$

The 'counts as' signature σ^{γ} contains all instances of the pow expressions. And instead of making them simple and inertial, in this example we will make them statically determined and false by default.

Version 1 Although it is a simple idea, revocation complicates the formulation substantially. So first we show a version without revocation, so as not to distract attention from the main points. We will discuss revocation separately after that.

Preliminary:

$$\begin{array}{l}\textsf{inertial } \mathit{current\_bidder},\ \mathit{current\_bid},\ \mathit{withdrawn}(X)\\ \textsf{nonexecutable } (X\ \mathit{signals}\ \alpha) \wedge (X\ \mathit{signals}\ \alpha')\end{array}$$

for all $\alpha \neq \alpha'$ ranging over $\mathit{open}\,(N)$, $\mathit{raise}\,(N)$, $\mathit{withdraw}$, revoke. The above allows for concurrent actions by different players. These can be eliminated if desired by replacing the nonexecutable laws above by adding

$$\begin{array}{l}\textsf{inertial } \mathit{current\_bidder},\ \mathit{current\_bid},\ \mathit{withdrawn}(X)\\ \textsf{nonexecutable } (X\ \mathit{signals}\ \alpha) \wedge (X'\ \mathit{signals}\ \alpha')\end{array}$$

for all $X \neq X'$ ranging over the names of the players.

Now we specify the powers. The nonexecutable laws below eliminate unsupported institutional actions.

$$\begin{array}{l}\textsf{nonexecutable } (X{:}\alpha) \wedge \neg(X\ \mathit{signals}\ \alpha)\\ \textsf{default } \neg\textsf{pow}(X,\alpha) \qquad \%\ \text{statically determined, and not inertial}\\ \textsf{pow}(X, \mathit{open}\,(N))\ \textsf{if}\\ \quad \mathit{player}(X)\ \wedge\ \neg\mathit{withdrawn}(X)\ \wedge\\ \quad \mathit{max\_raise} = \mathit{Max}\ \wedge\ 0 < N \leq \mathit{Max}\ \wedge\\ \quad \mathit{current\_bid} = \mathit{none}\\ \textsf{pow}(X, \mathit{raise}\,(N))\ \textsf{if}\\ \quad \mathit{player}(X)\ \wedge\ \neg\mathit{withdrawn}(X)\ \wedge\\ \quad \mathit{max\_raise} = \mathit{Max}\ \wedge\ 0 < N \leq \mathit{Max}\ \wedge\\ \quad \mathit{current\_bidder} \neq X\\ \textsf{pow}(X, \mathit{withdraw})\ \textsf{if}\\ \quad \mathit{player}(X)\ \wedge\ \neg\mathit{withdrawn}(X)\ \wedge\\ \quad \mathit{current\_bidder} \neq X\end{array}$$

Now we specify the effects of (successful, valid) actions:

$$\begin{array}{l}X{:}\mathit{open}\,(N)\ \textsf{causes}\ \mathit{current\_bidder} = X\\ X{:}\mathit{open}\,(N)\ \textsf{causes}\ \mathit{current\_bid} = N\\ X{:}\mathit{raise}\,(N)\ \textsf{causes}\ \mathit{current\_bidder} = X\\ X{:}\mathit{raise}\,(N)\ \textsf{causes}\ \mathit{current\_bid} = \mathit{Current} + N\ \textsf{if}\\ \qquad \mathit{current\_bid} = \mathit{Current}\\ X{:}\mathit{withdraw}\ \textsf{causes}\ \mathit{withdrawn}(X)\end{array}$$

The above uses some simple arithmetic, easily added. See e.g. the implementation provided as part of the Causal Calculator CCALC[4].

[4] http://www.cs.utexas.edu/users/tag/cc

Version 2 The complication with revocation is that we have to remember at least the last previous bid made so we know what to re-instate when revocation takes place. We do not want to implement a *stack* of bids—not in this example anyway. Notice that the extra complication is part of the temporal structure of the bidding process and not a result of the need to specify counts_as or pow. It is a limitation of $\mathcal{C}+$, and of transition systems in general, that a state is independent of the history of transitions by which it was reached.

So introduce another pair of inertial (multi-valued) fluent constants, *previous_bid* and *previous_bidder*. They are inertial because some actions (such as withdrawals) do not affect the current bid. A *raise* action now changes the value of *current_bid* and *current_bidder* as before, but first sets *previous_bid* and *previous_bidder*. In addition to the laws of version 1 we have:

$$X{:}\mathit{raise}\,(N) \text{ causes } \mathit{previous\_bidder} = Y \text{ if } \mathit{current\_bidder} = Y$$
$$X{:}\mathit{raise}\,(N) \text{ causes } \mathit{previous\_bid} = \mathit{Tot} \text{ if } \mathit{current\_bid} = \mathit{Tot}$$

where Y ranges over the names of the players and *Tot* ranges over integers in the range $1..M_{\max}$.

Revocation when it happens re-instates the *previous_bid* and *previous_bidder* as the *current_bid* and *current_bidder*:

$$X{:}\mathit{revoke} \text{ causes } \mathit{current\_bidder} = Y \text{ if } \mathit{previous\_bidder} = Y$$
$$X{:}\mathit{revoke} \text{ causes } \mathit{current\_bid} = \mathit{Tot} \text{ if } \mathit{previous\_bid} = \mathit{Tot}$$

Notice that this formulation also deals with revocations of the opening bid. In that case only the second of the clauses above has any effect: the first and third have no effect because there is no *previous_bid* following an opening bid.

It remains to specify when agent X is empowered to revoke. One way is to attempt to specify it by formulating appropriate combinations of *previous_bid* and *current_bid*. Clearer, and more natural, is to do it by specifying further effects of *open* , *raise* , and *revoke*, as follows:

$$X{:}\mathit{open}\,(N) \text{ causes } \mathsf{pow}(X, \mathit{revoke})$$
$$X{:}\mathit{raise}\,(N) \text{ causes } \mathsf{pow}(X, \mathit{revoke})$$
$$X{:}\mathit{raise}\,(N) \text{ causes } \neg\mathsf{pow}(Y, \mathit{revoke}) \text{ if } \mathsf{pow}(Y, \mathit{revoke})$$
$$X{:}\mathit{revoke} \text{ causes } \neg\mathsf{pow}(X, \mathit{revoke})$$

This completes the specification of the example.

To illustrate that instances of counts_as can sometimes be chained together, suppose that agent Bob is to act as a representative in the auction of another agent Charles. This arrangement might be expressed as follows:

$$\mathit{Bob\ signals\ raise}\,(N) \text{ counts\_as } \mathit{Charles\ signals\ raise}\,(N) \text{ if } \top$$

There are other forms of representation, also expressible in the language $(\mathcal{C}+)^+$. We do not attempt any systematic account of the possibilities here. Some suggestions along these lines, using a different formalism, are discussed for example in (Gelati et al., 2004).

Note also that actions can have institutional effects which are not of the 'power' kind. Suppose, for example, that Charles promises or is instructed that every

time he makes a bid, he must inform his superior, Clare. Here, a bid by Charles creates an obligation on him to perform another action. There is some ambiguity here about whether Charles is obliged to inform Clare of every attempted bid ('signals') or only of successful bids. The distinction is evidently expressible in the $(\mathcal{C}+)^+$ language.

Further structuring It would be advantageous to introduce some further structure in the specification of pow, for instance by separating out the time-dependent from the time-independent parts. More generally, we might separate the components that make a bid 'valid' (better perhaps: 'well formed') from the conditions defining pow and counts_as. The difficulty here is not in devising such representations but in fixing on terminology. Terms such as 'valid', 'well formed', 'inadmissible', 'improper', 'out of order', 'void', 'voidable' have specific meanings in certain contexts, but these contexts are relatively few, and the same meaning is not always given in each. There is unfortunately much scope for confusion here. It would be better to identify and fix on some neutral terms, but these are difficult to find. Some preliminary suggestions are provided in (Prakken, 1998; Prakken and Gordon, 1999).

Examples of what can be done with such formalisations are provided in (Artikis et al., 2002, 2003a,b). There, similar protocols are formalised using event calculus and/or the language $\mathcal{C}+$ as the action formalism and then subjected to various kinds of analysis. These examples can be reformulated using the additional resources of the language $(\mathcal{C}+)^+$ instead. (We do not present these reformulations here.) The advantage of using $(\mathcal{C}+)^+$ is that it provides a built-in treatment of counts_as and pow, and their properties, which otherwise have to anticipated and coded up explicitly as $\mathcal{C}+$ laws.

8.8 Unsupported institutional actions

We have chosen to leave the elimination of unsupported institutional actions to the formulation of action descriptions. A case can be made that this is unreasonable, that the elimination of unsupported institutional actions is fundamental to the notion of 'counts as' being modelled, and that this should therefore not be a feature of action descriptions but rather a fixed semantic feature of ATS structures. It is easy to formulate the required conditions. Where $\alpha_1 \,\mathsf{counts\_as}\, \beta,\ \ldots,\ \alpha_n \,\mathsf{counts\_as}\, \beta$ are all the constants of the form $\alpha \,\mathsf{counts\_as}\, \beta$ in the signature σ^γ, we require that every transition (s, ε, s') of the ATS has the property that $\varepsilon \not\models \beta \wedge \neg\alpha_1 \wedge \cdots \wedge \neg\alpha_n$, i.e., that $\varepsilon \models \beta \rightarrow (\alpha_1 \vee \cdots \vee \alpha_n)$. (It may seem that this is equivalent to the condition $\varepsilon \models \beta \leftrightarrow (\alpha_1 \vee \cdots \vee \alpha_n)$ but this is not so, since we require $\varepsilon \models \alpha_i \rightarrow \beta$ only when $\alpha_i \,\mathsf{counts\_as}\, \beta$ holds in state s of the transition (s, ε, s'), whereas the condition required to eliminate unsupported institutional actions depends only on the signature σ^γ and not on the counts_as atoms that actually hold in state s.)

However, this treatment would also pre-suppose an assumption of completeness in the specification of the σ^γ signature. It would assume that there are no other means of effecting the institutional action β besides those actions α_i that appear in $\alpha_i \,\mathsf{counts\_as}\, \beta$ constants in σ^γ. It is at least conceivable that this assumption would not always be desired, and in these circumstances transition labels/events

with unsupported institutional actions would be meaningful even if in general they are not. It is, in any case, very easy to ensure that an action description eliminates (undesired) unsupported institutional actions. To eliminate an unsupported institutional action β one includes in the action description a fluent dynamic law

$$\mathsf{nonexecutable}\ \beta \wedge \neg\alpha_1 \wedge \cdots \wedge \neg\alpha_n$$

where α_1 $\mathsf{counts\_as}$ β, $\ldots$, α_n $\mathsf{counts\_as}$ β are all the constants of the form α $\mathsf{counts\_as}$ β in the signature σ^γ. This guarantees that all transitions (s, ε, s') of the ATS defined satisfy the property $\varepsilon \models \beta \rightarrow (\alpha_1 \vee \cdots \vee \alpha_n)$, without introducing any new transitions. For these two reasons together (no assumptions of completeness in σ^γ, and the ease of adding the required $\mathsf{nonexecutable}$ laws in the action description where desired) we do not include the elimination of unsupported institutional actions as a fixed feature of the semantics.

Some care has to be taken when deciding on the 'counts as' signature σ^γ. Compare a signature containing the single constant α $\mathsf{counts\_as}$ $\beta_1 \wedge \beta_2$, on the one hand, with a signature containing the pair of constants α $\mathsf{counts\_as}$ β_1 and α $\mathsf{counts\_as}$ β_2, on the other. These have different effects. In both cases, a transition of type $\beta_1 \wedge \beta_2 \wedge \neg\alpha$ represents (assuming no other relevant $\mathsf{counts\_as}$ constants) an unsupported institutional action: $\beta_1 \wedge \beta_2$ is unsupported in the first case, and both β_1 and β_2 are unsupported in the second. However, in the first case a transition of type $\beta_1 \wedge \neg\beta_2 \wedge \neg\alpha$ does not represent an unsupported institutional action, whereas in the second case it does (β_1 is unsupported).

Implicit in this treatment of 'counts as' is that the conjunction $\beta_1 \wedge \beta_2$ in a constant α $\mathsf{counts\_as}$ $\beta_1 \wedge \beta_2$ represents a single action whose attributes are specified by $\beta_1 \wedge \beta_2$. In contrast, when $\beta_1 \wedge \beta_2$ is intended to represent the concurrent execution of two separate actions β_1 and β_2, two separate constants α $\mathsf{counts\_as}$ β_1 and α $\mathsf{counts\_as}$ β_2 should be employed.

As a further example, consider the difference in terms of unsupported institutional actions between a signature $S1$ containing two constants

$$\begin{array}{l} \alpha_1\ \mathsf{counts\_as}\ \beta \wedge \beta_1 \\ \alpha_2\ \mathsf{counts\_as}\ \beta \wedge \beta_2 \end{array} \qquad (S1)$$

and a signature $S2$ containing four separate constants

$$\begin{array}{ll} \alpha_1\ \mathsf{counts\_as}\ \beta & \alpha_1\ \mathsf{counts\_as}\ \beta_1 \\ \alpha_2\ \mathsf{counts\_as}\ \beta & \alpha_1\ \mathsf{counts\_as}\ \beta_2 \end{array} \qquad (S2)$$

In the case $S1$, β would typically be a conjunction of some common set of core attribute-values and β_1 and β_2 conjunctions of additional attribute-values to represent the two different types of β actions. In the case $S2$, β, β_1, and β_2 would each represent a particular kind of action, though of course there is nothing in the syntax of $\mathcal{C}+$ (or of $(\mathcal{C}+)^+$) that indicates the difference explicitly.

In the case $S1$, a transition has an unsupported institutional action when it satisfies $\beta \wedge \beta_1 \wedge \neg\alpha_1$ or $\beta \wedge \beta_2 \wedge \neg\alpha_2$; in the case $S2$ when it satisfies $\beta \wedge \neg\alpha_1 \wedge \neg\alpha_2$ or $\beta_1 \wedge \neg\alpha_1$ or $\beta_2 \wedge \neg\alpha_2$. One can see that if a transition has an unsupported institutional action with signature $S1$ then it also has an unsupported institutional action with signature $S2$, though not *vice-versa*. However, if every transition label/event also satisfies $\beta \leftrightarrow (\beta_1 \vee \beta_2)$, then the two are equivalent. So: suppose

that in $S1$ the conjunction β does represent some common set of core attribute values, and that there are exactly two kinds of β actions, one represented by the full set of attribute values $\beta \wedge \beta_1$, and the other represented by $\beta \wedge \beta_2$. Now suppose we make this explicit in the action description, e.g., by including the three fluent dynamic laws nonexecutable $\beta \wedge \neg\beta_1 \wedge \neg\beta_2$, nonexecutable $\beta_1 \wedge \neg\beta$, and nonexecutable $\beta_2 \wedge \neg\beta$. Then every transition label/event in the ATS defined satisfies $\beta \leftrightarrow (\beta_1 \vee \beta_2)$, and the signature $S1$ and $S2$ agree on the conditions for unsupported institutional actions.

There is clearly more to this. It may be worthwhile extending the syntax of $(\mathcal{C}+)^+$ (and $\mathcal{C}+$) to make explicit when a conjunction of action atoms is intended to represent the attributes of a single action and when it is intended to represent instead the concurrent execution of several distinct actions. One can see how this might be done, by having a separate category of action-attribute constants in the signature σ^{a}, for instance. We leave the details for future work.

8.9 Discussion: Possible strengthenings

We turn now to the possibility of strengthening the 'counts as' relation $C^+_{\mathcal{I}}$ to a transitive relation or to an ordering on (the special category of 'institutionally significant') transition types, as outlined in the earlier discussion in Section 8.1 on comparisons with 'counts as' conditionals in (Jones and Sergot, 1996) and 'conventional generation' in (Goldman, 1970). Since reflexivity of 'counts as' is of negligible practical interest, there are two main issues: the difficulty of implementing the transitivity conditions in the translation from $(\mathcal{C}+)^+$ to causal theories, and the choice of the most appropriate formulation of transitivity. We limit ourselves to some remarks about possibilities and the issues arising, and leave detailed investigation to a future version.

We note first that the constraint $(\mathrm{C}^+_\varepsilon)$ in every ATS already gives a kind of 'pseudo-transitivity' of $C^+_{\mathcal{I}}$. Expressed in terms of $\tau_R(s, X)$, the set of transitions of type X executable in state s, $(\mathrm{C}^+_\varepsilon)$ takes the form (for all transition types X, Y and all states s):

$$\text{if } C^+_{\mathcal{I}}(s, X, Y) \text{ then } \tau_R(s, X) \subseteq \tau_R(s, Y)$$

from which follows the 'pseudo-transitivity' property (for all transition types X, Y, Z and all states s):

$$\text{if } C^+_{\mathcal{I}}(s, X, Y) \text{ and } C^+_{\mathcal{I}}(s, X, Z) \text{ then } \tau_R(s, X) \subseteq \tau_R(s, Z)$$

as well as a (very weak) kind of 'pseudo-anti-symmetry'

$$\text{if } C^+_{\mathcal{I}}(s, X, Y) \text{ and } C^+_{\mathcal{I}}(s, Y, X) \text{ then } \tau_R(s, X) = \tau_R(s, Y)$$

and a (trivial) 'pseudo-reflexivity'

$$\text{if } C^+_{\mathcal{I}}(s, X, X) \text{ then } \tau_R(s, X) = \tau_R(s, X)$$

But suppose we wished to strengthen these properties as suggested in Section 8.1, to

$$C^+_{\mathcal{I}}(s, X, X) \qquad (\mathrm{C}^+_{\mathrm{refl}})$$

$$\text{if } C^+_{\mathcal{I}}(s, X, Y) \text{ and } C^+_{\mathcal{I}}(s, Y, Z) \text{ then } C^+_{\mathcal{I}}(s, X, Z) \qquad (\mathrm{C}^+_{\mathrm{trans}})$$

$$\text{if } C^+_{\mathcal{I}}(s, X, Y) \text{ and } C^+_{\mathcal{I}}(s, Y, X) \text{ then } X = Y \qquad (\mathrm{C}^+_{\text{a-symm}})$$

for all states s and for all transition types $X, Y, Z \subseteq \mathbf{A}_{\mathcal{I}}$, where, recall, $\mathbf{A}_{\mathcal{I}}$ denotes the transition types of special conventional significance in institution $\mathcal{I}$, defined in this paper as $\mathbf{A}_{\mathcal{I}} =_{\text{def}} \{X \subseteq \mathbf{A} \mid X = \|\alpha\| \text{ or } X = \|\beta\| \text{ for some constant } \alpha \,\mathsf{counts\_as}\, \beta \text{ in } \sigma^{\gamma}\}$.

The first step is easy. It is easy to adjust the definition of the ATS defined by a $(\mathcal{C}+)^+$ action description (Definition 13) so that the 'counts as' relation has the desired additional properties. Let D be an action description of $(\mathcal{C}+)^+$ of signature $(\sigma^{\text{f}}, \sigma^{\gamma}, \sigma^{\text{a}})$. The *super-augmented* transition system defined by D is the structure

$$\langle \sigma^{\text{f}}, \sigma^{\gamma}, S, \mathbf{A}, R, C_D^{++} \rangle$$

where $\langle \sigma^{\text{f}}, \sigma^{\gamma}, S, \mathbf{A}, R, C_D^{+} \rangle$ is the ATS defined by D according to Definition 13 and where the new 'counts as' relation C_D^{++} is defined to be the 'reflexive transitive closure' of C_D^{+}, or more precisely, the smallest relation containing C_D^{+} that satisfies the conditions ($\text{C}^{+}_{\text{refl}}$) and ($\text{C}^{+}_{\text{trans}}$). If we further specify that C_D^{+} must be cycle-free, then C_D^{++} also satisfies ($\text{C}^{+}_{\text{a-symm}}$), and so is an ordering on the 'institutionally significant' transition types $\mathbf{A}_{\mathcal{I}}$ in the sense discussed in Section 8.1. It is easy to check that the condition ($\text{C}^{+}_{\varepsilon}$) is satisfied by C_D^{++} if it is satisfied by C_D^{+}, and so the structure defined in this way is an ATS.

So far so good. The problem comes in trying to find a causal theory, or a translation to a $\mathcal{C}+$ action description, that will encode the paths of the 'super-augmented' transition sytem, and specifically additional causal laws that will construct C_D^{++} as the 'reflexive transitive' closure of C_D^{+}.

It might appear that ($\text{C}^{+}_{\text{refl}}$) and ($\text{C}^{+}_{\text{trans}}$) can be captured straightforwardly by including further laws of the form:

$$\alpha \,\mathsf{counts\_as}\, \alpha \;\mathsf{if}\; \top \tag{12}$$

for every constant $\alpha \,\mathsf{counts\_as}\, \alpha$ in σ^{γ},

$$\alpha \,\mathsf{counts\_as}\, \beta \;\; \mathsf{if} \;\; \alpha \,\mathsf{counts\_as}\, \beta' \;\wedge\; \beta' \,\mathsf{counts\_as}\, \beta \tag{13}$$

for all constants $\alpha \,\mathsf{counts\_as}\, \beta$, $\alpha \,\mathsf{counts\_as}\, \beta'$, and $\beta' \,\mathsf{counts\_as}\, \beta$ in σ^{γ}, and

$$\mathsf{default}\; \neg (\alpha \,\mathsf{counts\_as}\, \beta) \tag{14}$$

for every constant $\alpha \,\mathsf{counts\_as}\, \beta$ in σ^{γ} to capture the closure condition.

But this does not work. As pointed out in (Giunchiglia et al., 2004, sect.7.2) expressing the reflexive transitive closure of a relation in the language of causal theories is not straightforward. For example, at first sight the causal laws

$$q(x,x) \Leftarrow \top, \quad q(x,y) \Leftarrow p(x,z) \wedge q(z,y), \quad \neg q(x,y) \Leftarrow \neg q(x,y)$$

may appear to express that q is the reflexive transitive closure of p. But they do not: there *is* a model of the causal theory in which q is the reflexive transitive closure of p but there are also other models in which q, though reflexive and transitive, contains more than the reflexive transitive closure of p. Consider $\{p(1,1), p(2,2)\}$. The reflexive transitive closure is $\{q(1,1), q(2,2)\}$, represented by the model $\{q(1,1), q(2,2), \neg q(1,2), \neg q(2,1)\}$, but the causal theory also has three other models: $\{q(1,1), q(2,2), q(1,2), \neg q(2,1)\}$, $\{q(1,1), q(2,2), \neg q(1,2),$

$q(2,1)\}$, and $\{q(1,1), q(2,2), q(1,2), q(2,1)\}$. It is perhaps easier to see why this is so by comparison with symmetric closure. The causal laws

$$q(x,y) \Leftarrow p(x,y), \quad q(x,y) \Leftarrow q(y,x), \quad \neg q(x,y) \Leftarrow \neg q(x,y)$$

similarly fail to express that q is the symmetric closure of p. In the example $\{p(1,1), p(2,2)\}$, the causal theory will contain the instances $q(1,2) \Leftarrow q(2,1)$, $q(2,1) \Leftarrow q(1,2)$, $\neg q(1,2) \Leftarrow \neg q(2,1)$, and $\neg q(2,1) \Leftarrow \neg q(1,2)$ and it is easy to see that we get two models: $\{q(1,1), q(2,2), \neg q(1,2), \neg q(2,1)\}$ as intended, but also $\{q(1,1), q(2,2), q(1,2), q(2,1)\}$.

It is not clear how this problem can be resolved. Since we view translations to causal theories as essentially an implementation issue, one possibility is to place the burden elsewhere in the implementation, that is to say, not in the component that constructs causal theories from $(\mathcal{C}+)^+$ action descriptions but in the component that evaluates queries on the causal theories so constructed. To evaluate formulas of some query language on the 'super-augmented' ATS defined by a $(\mathcal{C}+)^+$ action description D, we construct the causal theory ${\Gamma^+}_m^D = \Gamma_m^{D \cup \delta_D \cup \delta_{\sigma^\gamma}}$ representing the (regular) ATS defined by D, which encodes the 'counts as' relation C_D^+, but arrange that the associated query evaluator evaluates formulas containing $\mathsf{counts\_as}_{\mathcal{I}}$ expressions not on C_D^+ but on the relation C_D^{++}. The process of constructing (the relevant instances of) C_D^{++} from (the relevant instances of) C_D^+ is the job of the query evaluator, not part of the translation from D to ${\Gamma^+}_m^D$.

This is not difficult to implement, but it is not satisfactory either. It would mean that $\mathsf{counts\_as}$ constants in the $(\mathcal{C}+)^+$ language do not behave in the same way, do not have the same properties as, the corresponding $\mathsf{counts\_as}_{\mathcal{I}}$ expressions in the query language. One could say that this does not matter, that $\mathsf{counts\_as}$ constants in $(\mathcal{C}+)^+$ are used only to *define* the relations C_D^+ and (indirectly) C_D^{++}, and that the semantics of $(\mathcal{C}+)^+$ is unambiguous in this respect: one should not ascribe more properties to $\mathsf{counts\_as}$ constants than are stated by the semantics. But still: it is only natural to assume that $\mathsf{counts\_as}$ constants in the $(\mathcal{C}+)^+$ language will behave in the same way as the relations they define, and it is at best confusing if it is otherwise. For instance, suppose we write $a\ \mathsf{counts\_as}\ b\ \mathsf{if}\ \top$ in a $(\mathcal{C}+)^+$ action description, and also a fluent dynamic law $\alpha\ \mathsf{causes}\ (b\ \mathsf{counts\_as}\ c)$ (say). It would be entirely reasonable to suppose, given the intended (stronger, transitive) reading of $\mathsf{counts\_as}$, that in any state after an α transition, $a\ \mathsf{counts\_as}\ c$ will hold. But it does not—this kind of inference is not supported by the language $(\mathcal{C}+)^+$ itself. In the existing version, only the weaker 'pseudo-transitivity' is present: after an α transition, every transition of type a will be a transition of type c, but this does not imply that the constant $a\ \mathsf{counts\_as}\ c$ will be true. In the query language, in contrast, $a\mathsf{counts\_as}_{\mathcal{I}} c$ *would* be true, and this sort of mis-match is clearly to be avoided.

Implementation issues aside, there are in any case some other possible formulations of transitivity that can be considered as alternatives to $(\mathrm{C}^+_{\mathrm{trans}})$, and which remain to be investigated. Suppose for example that α counts as $\beta_1 \wedge \cdots \wedge \beta_j \wedge \beta_{j+1} \wedge \cdots \wedge \beta_n$, and that $\beta_1 \wedge \cdots \wedge \beta_j$ counts as α'. Would we want to conclude that α counts as α'? An affirmative suggests we should look at a stronger type of transitivity, of the form

$$C_{\mathcal{I}}^+(s, X, Y) \text{ and } C_{\mathcal{I}}^+(s, Y', Z) \text{ implies } C_{\mathcal{I}}^+(s, X, Z), \text{ if } Y \subseteq Y'$$

or possibly, of the even stronger form

$$C^+_{\mathcal{I}}(s, X, Y) \text{ and } C^+_{\mathcal{I}}(s, Y', Z) \text{ implies } C^+_{\mathcal{I}}(s, X, Z), \text{ if } \tau_R(s, Y) \subseteq \tau_R(s, Y')$$

which depends on the *actual* transitions R of an ATS and not just on truth-functional properties of action formulas. These, and other forms, remain to be investigated, and are not included in the present version of $(\mathcal{C}+)^+$.

We do not see this as an important limitation of $(\mathcal{C}+)^+$. The significance of the very restricted form of reflexivity is negligible. Transitivity of 'counts as' is adopted in (Jones and Sergot, 1996) in the absence of convincing counter-examples to the contrary, but is not a critical feature of the characterisation adopted there, and neither is it here.

8.10 Defeasibility of 'counts as'

There is one further remark that we wish to make, concerning the defeasibility of 'counts as'. We can expect that in many practical applications the laws specifying 'counts as' relationships will be formulated most naturally as general default rules that are subject to implicit exceptions, exceptions to exceptions, and so on. We do not see anything particularly unusual or problematic about the formalisation of such rules in $(\mathcal{C}+)^+$. $(\mathcal{C}+)^+$ inherits from $\mathcal{C}+$ and the formalism of causal theories a general treatment of non-monotonic inference, and can be used to formulate defeasible general rule and exception structures using devices familiar from the literature on knowledge representation. (See (Giunchiglia et al., 2004, Sect. 7) for a comparison of causal theories with other well-known formalisations of non-monotonic reasoning.)

It may seem however that there is a different, more fundamental, kind of defeasibility inherent in the treatment of 'counts as', and this deserves some comment.

It is a cornerstone of the treatment given here of 'counts as' relations that when α counts_as β then any transition of type α is also a transition of type β. This is expressed by the constraint (C^+_ε) required of an ATS. The effect is that when α counts_as β holds in a state s there can be no transitions of type $\alpha \wedge \neg\beta$ from the state s. The same constraint is implicit in formalisations of 'counts as' in other action formalisms, such as (Jones and Sergot, 1996), where it emerges in discussion of 'exercise of power'.

It might appear that (C^+_ε) as it stands is too strong. Suppose, for example, that α counts_as β and that β causes F. F holds in all states following the occurrence of a transition of type β; and since all occurrences of type α are also occurrences of type β, we get in effect that α 'causes' F. This is as intended: we want to be able to infer that actions α have effects by virtue of being actions of type α and that they also have effects by virtue of being counted, in institution $\mathcal{I}$, as actions of type β. By the same argument, when α counts_as β and β is non-executable, then α is also non-executable. For if there are no transitions of type β in some given state s, and all transitions of type α are also transitions of type β, then there can be no transitions of type α in state s either.

But suppose now that *raise_hand* counts_as *start_race*. And suppose further that *start_race* is non-executable in some state—the runners are not ready, the track is too wet, or any number of other conditions hold. We must infer, as above, that in these circumstances *raise_hand* is also non-executable. It seems that

whenever the race is unable to start, the starter is unable to raise his hand, in the sense that his action of doing so is non-executable.

Now there is nothing strange or problematic about this inference if the purpose of the action description is to act as a system *specification*. The fact that there are circumstances in which raising a hand is non-executable when we expect that it should be executable indicates that there is a flaw in the specification. The provision of computational tools to help identify such flaws is one of the main aims of the development.

But what if the purpose of the action description is to provide an accurate formal model of some real-world fragment? Just because a race is not ready to start we do not infer that the starter is (physically) unable to raise his hand. It might seem that some adjustment is necessary, and that this adjustment must concern condition (C_ε^+). Perhaps the constraint that when α counts_as β, *every* transition of type α is a transition of type β is too strong, and condition (C_ε^+) should be made *defeasible* and subject to exceptions in some way. This seems to be the position adopted by e.g. (Gelati et al., 2002, 2004) who have expressed the view (in a different formal framework) that the 'counts as' connection is inherently defeasible.

It seems to us that this is the wrong way of looking at the problem. It is not the condition (C_ε^+) that needs to be made defeasible but the formalisation of what counts as what. In the example, if starting the race is non-executable unless circumstances G hold, then it is simply not true to say that raising a hand counts as starting the race; raising a hand counts as starting the race if G holds, but not otherwise.

Schematically, we have α counts_as β; β causes F, but when $\neg F$ holds, β is non-executable, and so too is α. However, α counts_as β is usually conditional on some other factual circumstances: schematically, the pattern is α counts_as β if G and β causes F. When $\neg F$ holds, β is non-executable, and α is non-executable when $G \wedge \neg F$ holds. But it will often be the case that $G \wedge \neg F$ can never be true because of other features of the specification or of the real-world fragment being modelled. One purpose of a formalisation is precisely to detect such features.

In summary: it is not the condition (C_ε^+) that needs to be made defeasible. Viewed as a kind of conditional, α counts_as β is not a *defeasible* conditional, though in practice the laws defining when α counts_as β holds will often be naturally formulated as defeasible general rules subject to exceptions.

9 The language $(\mathcal{C}+)^{++}$: Permission

We now consider a second extension to the language $\mathcal{C}+$, to provide a means of specifying the 'permitted', 'acceptable', 'ideal', 'legal' states and the 'permitted', 'acceptable', 'ideal', 'legal' actions and transition paths. This extension is independent of the extension to accommodate 'counts as'/conventional generation, but since we will usually want both features in the intended applications we present it here as an extension to the language $(\mathcal{C}+)^+$ of the previous section.

The semantic structure is a 'coloured ATS' which is a structure of the form:

$$\langle \sigma^{\mathrm{f}}, \sigma^{\gamma}, S, \mathbf{A}, R, C_{\mathcal{I}}^{+}, S_{\mathrm{green}}, R_{\mathrm{green}} \rangle$$

(or a 'coloured LTS' of the form

$$\langle \sigma^{\mathrm{f}}, S, \mathbf{A}, R, S_{\mathrm{green}}, R_{\mathrm{green}} \rangle$$

if we do not need the 'counts as' features). The two new components are

- $S_{\mathrm{green}} \subseteq S$, the set of 'permitted' ('acceptable', 'ideal', 'legal') states—we call S_{green} the 'green' states of the system;
- $R_{\mathrm{green}} \subseteq R$, the set of 'permitted' ('acceptable', 'ideal', 'legal') transitions—we call R_{green} the 'green' transitions of the system.

We refer to the complements $S - S_{\mathrm{green}}$ and $R - R_{\mathrm{green}}$ as the 'red states' and 'red transitions', respectively. It is also possible to consider a more elaborate structure, of *partially coloured* transition systems in which states and transitions can be green, red, or uncoloured, but we shall not present that version here.

Note that $R_{\mathrm{green}} \subseteq R$ means that we build in the property that 'permission' implies 'can'. We impose the following additional constraint:

- if $(s, \varepsilon, s') \in R_{\mathrm{green}}$ and $s \in S_{\mathrm{green}}$ then $s' \in S_{\mathrm{green}}$.

which we refer to as the *green-green-green* constraint. Performing a permitted (green) action (transition) in a permitted (green) state must always lead to a permitted (green) state. All other possible combinations of green/red states and green/red transitions are allowed. In particular, and *contra* the assumptions underpinning John-Jules Meyer's (1988) construction of 'dynamic deontic logic', a non-permitted (red) transition can result in a permitted (green) state. Similarly, it is easy to devise examples in which a permitted (green) transition can lead to a non-permitted (red) state. Some illustrations will arise in the examples to be considered later in the section. The only combination that cannot occur is the one eliminated by the 'green-green-green' constraint: a permitted (green) transition from a permitted (green) state cannot lead to a non-permitted (red) state.

Semantic devices such as S_{green} and R_{green} are familiar in the field of deontic logic. For example, Carmo and Jones (1996) employ a similar structure (though not for the purpose of constructing a language for defining them, as we attempt here) which has ideal/sub-ideal states and ideal/sub-ideal transitions (unlabelled). van der Meyden's (1996) 'dynamic logic of permission' employs a structure in which transitions, but not states, are classified as 'permitted/non-permitted'. van der Meyden's version was constructed as a response to problems

of Meyer's 'dynamic deontic logic' (Meyer, 1988) which classifies transitions as 'permitted/non-permitted' by reference only to the state resulting from a transition. 'Deontic interpreted systems' (Lomuscio and Sergot, 2003) classify states as 'green'/'red', where these states have further internal structure to model the local states of agents in a multi-agent context. In all of these examples (and others) the task has been to find axiomatisations of such structures in one form of deontic logic or another. Here we are concerned with a different task, that of devising a language for *defining* coloured transition systems of the form described above.

9.1 The language $(\mathcal{C}+)^{++}$

The language $(\mathcal{C}+)^{++}$ extends the language $(\mathcal{C}+)^{+}$ with two new forms of laws.

- A *state permission law* (or rule) is an expression of the form

$$\textsf{not-permitted } F \tag{15}$$

  where F is a fluent formula (i.e., a formula of signature σ^{f}).
- An *action permission law* (or rule) is an expression of the form

$$\textsf{not-permitted } \alpha \textsf{ if } \psi \tag{16}$$

  where α is an action formula (i.e., a formula of signature σ^{a}) and ψ is a formula of signature $\sigma^{\mathrm{f}} \cup \sigma^{\mathrm{a}}$. $\textsf{not-permitted } \alpha$ is shorthand for $\textsf{not-permitted } \alpha \textsf{ if } \top$.
  It is also convenient to allow two variants of rule forms (15) and (16): $\textsf{oblig } F$ is an abbreviation for $\textsf{not-permitted } \neg F$ and $\textsf{oblig } \alpha$ is an abbreviation for $\textsf{not-permitted } \neg\alpha$.

Informally, in the transition system defined by an action description D, a state s is red whenever $s \models F$ for any state permission law $\textsf{not-permitted } F$. All other states are green by default. A transition (s, ε, s') is red whenever $s \cup \varepsilon \models \psi$ and $\varepsilon \models \alpha$ for any action permission law $\textsf{not-permitted } \alpha \textsf{ if } F \textsf{ after } \psi$. All other transitions are green, *subject to* the 'green-green-green' constraint which may impose further conditions on the possible colouring of a given transition. Examples will follow shortly to illustrate the effects of the 'green-green-green' constraint. It is possible to construct variants of the language with the defaults working the other way round (that is, a variant where the permission laws specify what is permitted (green), with other states and transitions not permitted (red) by default), or more elaborate forms allowing defaults to be mixed, but these all turn out to be more awkward and rather unnatural. We will stick to one basic form in this present account.

Let D be an action description of $(\mathcal{C}+)^{++}$. D_{basic} refers to the subset of laws of D that are also laws of $(\mathcal{C}+)^{+}$ (and hence also laws of $\mathcal{C}+$). The coloured ATS defined by D has the augmented states S and transitions R that are defined by its $(\mathcal{C}+)^{+}$ component D_{basic}. In the case where we are not using the 'counts as' features, the coloured LTS defined by D has the states S and transitions R of the labelled transition system defined by the $\mathcal{C}+$ action description D_{basic}. In both cases, coloured ATS and coloured LTS, the green states S_{green} and green transitions R_{green} are defined as follows:

$$S_{\mathrm{green}} =_{\mathrm{def}} S - S_{\mathrm{red}} \qquad R_{\mathrm{green}} =_{\mathrm{def}} R - R_{\mathrm{red}}$$

where

$$
\begin{aligned}
S_{\mathrm{red}} =_{\mathrm{def}} & \{s \mid s \models F \text{ for some state permission law } \mathsf{not\text{-}permitted}\ F \text{ in } D\} \\
R_{\mathrm{red}} =_{\mathrm{def}} & \{(s, \varepsilon, s') \mid s \cup \varepsilon \models \psi,\ \varepsilon \models \alpha,\ s' \models F \text{ for some action} \\
& \qquad\qquad \text{permission law } \mathsf{not\text{-}permitted}\ \alpha\ \mathsf{if}\ F\ \mathsf{after}\ \psi \text{ in } D\} \\
& \cup\ \{(s, \varepsilon, s') \mid s \in S_{\mathrm{green}} \text{ and } s' \notin S_{\mathrm{green}}\}
\end{aligned}
$$

The last component of the R_{red} definition ensures that the 'green-green-green' constraint is satisfied.

9.2 Example

Suppose it is forbidden for a man and a woman to be alone together in a room. Let a and b be men and c be a woman, and suppose they inhabit a world in which there is one dwelling consisting of two rooms with a connecting internal door and one external door, as depicted in Figure 9.1. For the sake of simplicity, suppose further that exactly one of the three persons a, b and c moves from one room to another at any one time, and ignore the possibility that a, b or c move out of the dwelling or other persons move in from the outside. These simplifications are imposed merely to keep the diagrams of the resulting transition system manageable. The example works just as well without the simplifications but the diagrams become too large to exhibit here.

Let fluents $loc(x) = p$ represent the location of person x, for x ranging over a, b, and c, and p ranging over the locations 'left' and 'right' (say). Let $move(x) = p$ represent the action in which x moves to location p. The action description for the example specifies that the $loc(x)$ fluents are inertial, that the $move(x)$ actions are exogenous, that exactly one $move(x)$ action takes place at any transition, that $move(x) = p$ is nonexecutable when $loc(x) = p$, and that $move(x) = p$ 'causes' $loc(x) = p$. These parts are just as for the language $\mathcal{C}+$; the details are straightforward and omitted. The state permission laws for the example may be stated in various ways, of which one simple formulation is as follows (where p ranges over the two locations 'left' and 'right'):

$$
\begin{aligned}
&\mathsf{not\text{-}permitted}\ loc(a) = p \wedge loc(c) = p \wedge \neg loc(b) = p \\
&\mathsf{not\text{-}permitted}\ loc(b) = p \wedge loc(c) = p \wedge \neg loc(a) = p
\end{aligned}
$$

There are more general and more elegant ways of formulating the required laws; this simple form is adequate for present purposes and does not distract from the main point of the example.

The transition system defined by this action description is shown in Figure 9.1, first with just the states coloured as determined by the state permission laws, and then with the colours of the transitions shown as determined by the 'green-green-green' constraint.

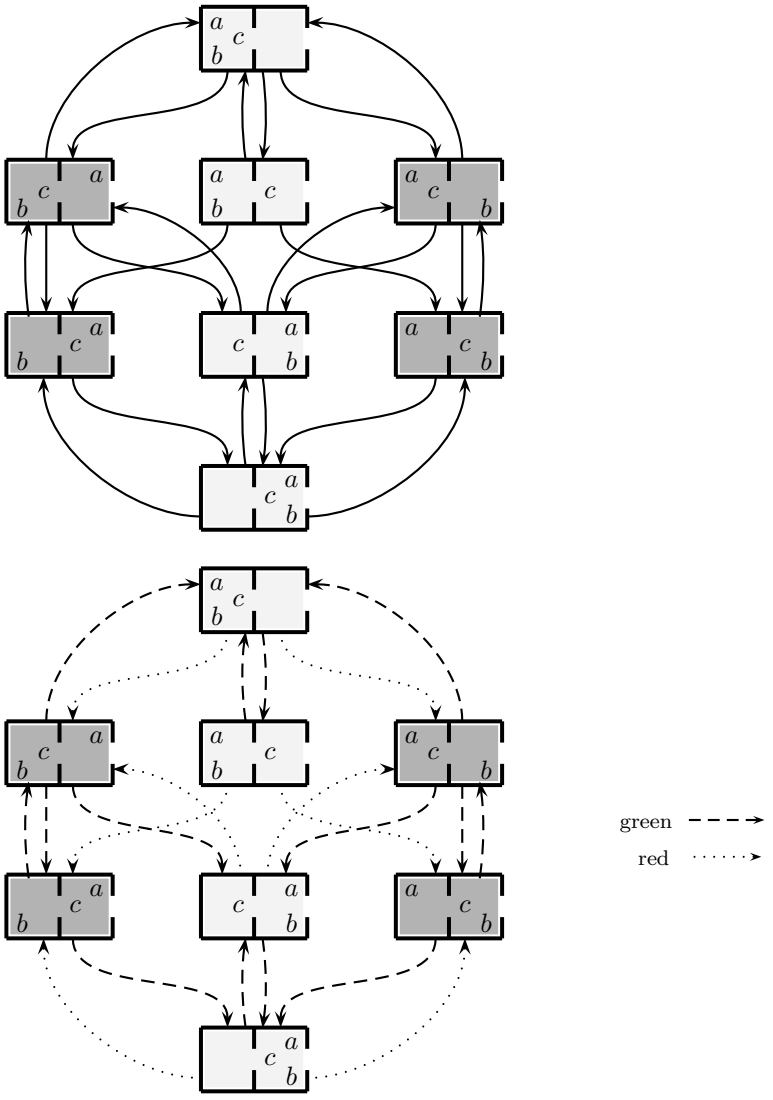


Figure 9.1: The transition system for the connecting rooms example. The top picture shows the colouring of the states: the darker shade indicates the red states as determined by the state permission laws, the others are green by default. The bottom diagram shows the colouring of the transitions, as determined by the 'green-green-green' constraint.

9.3 Translation to causal theories

The translation of an action description D in $(\mathcal{C}+)^{++}$ to a causal theory Γ_m^D is exactly as for the translation of $(\mathcal{C}+)^+$ theories, with the following extension to deal with the state and action permission laws.

Let status and trans be distinguished fluent and action constants, respectively, in the signature of Γ_m^D, both with possible values green and red. They will be used to represent the colour of a state and the colour of a transition, respectively.

For every state permission law not-permitted F and time index $i \in 0\,..\,m$, include in Γ_m^D a causal rule of the form

$$\mathsf{status}[i] = \mathsf{red} \Leftarrow F[i] \tag{17}$$

and for every $i \in 0\,..\,m$, a causal rule of the form

$$\mathsf{status}[i] = \mathsf{green} \Leftarrow \mathsf{status}[i] = \mathsf{green} \tag{18}$$

to specify the default colour of a state. A state permission law of the form oblig F produces causal rules of the form

$$\mathsf{status}[i] = \mathsf{red} \Leftarrow \neg F[i]$$

For every action permission law not-permitted α if F after ψ and time index $i \in 0\,..\,m{-}1$, include in Γ_m^D a causal rule of the form

$$\mathsf{trans}[i] = \mathsf{red} \Leftarrow F[i{+}1] \wedge \alpha[i] \wedge \psi[i] \tag{19}$$

and for every $i \in 0\,..\,m{-}1$, a causal rule of the form

$$\mathsf{trans}[i] = \mathsf{green} \Leftarrow \mathsf{trans}[i] = \mathsf{green} \tag{20}$$

to specify the default colour of a transition. An action permission law of the form oblig α if F after ψ produces causal rules of the form

$$\mathsf{trans}[i] = \mathsf{red} \Leftarrow F[i{+}1] \wedge \neg\alpha[i] \wedge \psi[i]$$

Finally, to capture the 'green-green-green' constraint, include for every $i \in 0\,..\,m{-}1$ a causal rule of the form

$$\mathsf{trans}[i] = \mathsf{red} \Leftarrow \mathsf{status}[i] = \mathsf{green} \wedge \mathsf{status}[i{+}1] = \mathsf{red} \tag{21}$$

It is straightforward to show that models of the causal theory Γ_m^D correspond to all paths of length m through the coloured transition system defined by D, where the fluent constant status and the action constant trans encode the colours of the states and transitions, respectively.

Note that when an action description D in $(\mathcal{C}+)^{++}$ is definite, it translates to a causal theory Γ_m^D which is also definite.

Causal rules (17), (18) and (20) can be expressed in the language $\mathcal{C}+$, that is to say, state permission laws and the default colouring of states and transitions can be expressed as laws in $\mathcal{C}+$, with status and trans treated as ordinary fluent and action constants. The causal rules (19) and (21), however, corresponding to action permission laws and the green-green-green constraint, cannot be expressed

in $\mathcal{C}+$. The form of these rules, where there is a time index in the antecedent greater than the time index in the consequent, conflicts with the very strong reading of 'causes' that underpins the language $\mathcal{C}+$. Rules corresponding to (19) and (21) are deliberately excluded from the language $\mathcal{C}+$, by design. $\mathcal{C}+$ could express a restricted form of action permission law, of the form

$$\mathsf{not\text{-}permitted}\ \alpha\ \mathsf{if}\ \top\ \mathsf{after}\ \psi \tag{22}$$

since this translates to the causal rule

$$\mathsf{trans}[i] = \mathsf{red} \Leftarrow \alpha[i] \wedge \psi[i]$$

which is the translated form of an action dynamic law of $\mathcal{C}+$. However, there is no similar simplification that allows the 'green-green-green' constraint to be expressed satisfactorily as a law in $\mathcal{C}+$. $\mathcal{C}+$ can express:

$$\bot\ \mathsf{if}\ \mathsf{status} = \mathsf{red}\ \mathsf{after}\ \mathsf{trans} = \mathsf{green} \wedge \mathsf{status} = \mathsf{green}$$

which translates to causal rules:

$$\bot \Leftarrow \mathsf{status}[i{+}1] = \mathsf{red} \wedge \mathsf{trans}[i] = \mathsf{green} \wedge \mathsf{status}[i] = \mathsf{green} \tag{23}$$

and it can express

$$\mathsf{status} = \mathsf{green}\ \mathsf{after}\ \mathsf{trans} = \mathsf{green} \wedge \mathsf{status} = \mathsf{green}$$

which translates to causal rules:

$$\mathsf{status}[i{+}1] = \mathsf{green} \Leftarrow \mathsf{trans}[i] = \mathsf{green} \wedge \mathsf{status}[i] = \mathsf{green} \tag{24}$$

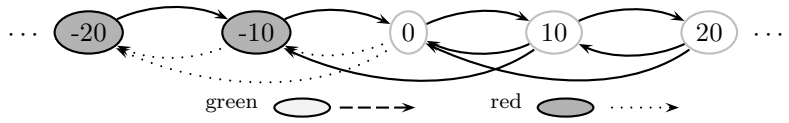
but both of these give quite the wrong effect. Both can be interpreted as defining transition systems. But although these transition systems satisfy the green-green-green constraint, they give the wrong interaction between state and action permission laws and the associated defaults.

The following example provides an illustration.

9.4 The 'green-green-green' constraint: Example

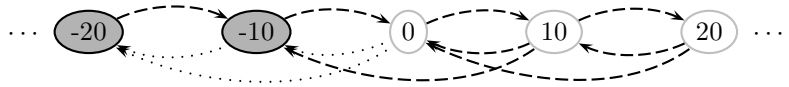
Consider a simple example of a bank (with one customer): states of the bank are represented by the value of a single fluent constant *balance* which represents the current balance of the customer's account. Suppose values of *balance* are multiples (including negative multiples) of 10 euro. There are three actions that the customer can perform: withdraw 10 euro, withdraw 20 euro, deposit 10 euro. A state is forbidden (not permitted, red) if the balance is less than 0. Suppose further, for the sake of the example, that a withdrawal (of any amount) is forbidden (red) when the balance is zero or negative. A suitable action description for this example in $(\mathcal{C}+)^{++}$ is very straightforward; we omit the details.

The transition system, colouring first only the states and transitions that are explictly specified to be red, is as follows:



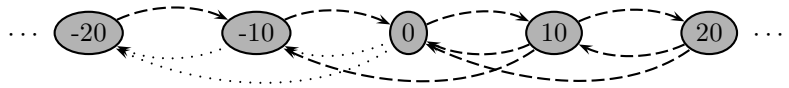
The labels on transitions are obvious from context so we omit them for clarity.

Suppose first that we apply the defaults and the green-green-green constraint in a different order to that specified in Section 9.3 above: suppose we first assume that all transitions not explicitly forbidden (red) are permitted (green), then apply the green-green-green constraint, and only then the default that states not red are green. At the first step we obtain the following partially coloured transition system:



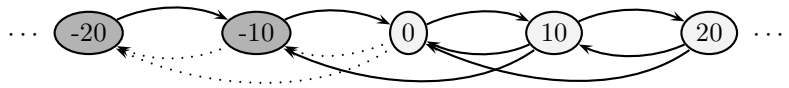
Now applying the green-green-green constraint forces us to conclude that the state 10 is red. This is because the transition from 10 to -10 is green, and the state -10 is red, so the state 10 cannot be green; it must be red.

Furthermore, state 0 must now also be red, because there is a green transition from 0 to a red state, 10. And the state 20 must be red too, because there is a green transition from a red state, 10, to 20. And so on, for all states greater than 20. We are left with the following picture.

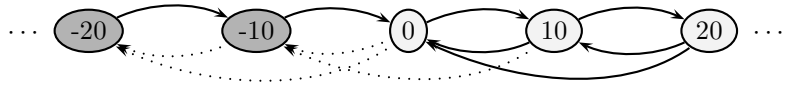


The result is surely quite counter-intuitive: we conclude that all balances, positive as well as negative, are not permitted, which is surely wrong.

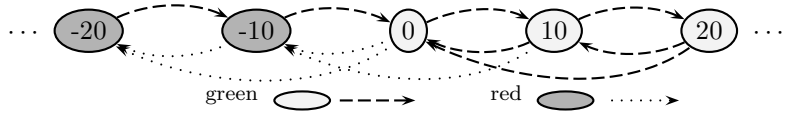
Contrast this with what is obtained by treating constraints and defaults in the manner specified in Section 9.3. First, states that are not explicitly red are green by default:



Now apply the green-green-green constraint: the transition from 10 to -10 must be red because 10 is a permitted (green) state while -10 is a forbidden (red) one.



Finally all other transitions are green by default, leaving the transition system coloured as follows.



And surely this is what we would expect: 10, 20, ... are all permitted states, and it is the transition from 10 to −10, i.e., the withdrawal of 20 euro when the balance is at 10 euro, that is not permitted.

Notice also that in this example, *contra* Meyer's definition of forbidden actions in 'dynamic deontic logic' (Meyer, 1988), there is a green transition (from −20 to −10) which results in a red state. And it is surely right that this transition should be green: it is surely permitted, given the permission laws stated above and the declared reading of defaults, that the customer is permitted to deposit 10 euro even though this still leaves his account in the negative. Although not shown in this example, it is also easy to find cases where a forbidden (red) transition can lead to a green (permitted) state. Suppose, for instance, that the bank decides to specify that withdrawals of less than 20 euro are no longer permitted (because of the administrative inconvenience, say): such withdrawals are still possible (they are not 'nonexecutable') but from the bank's point of view they are undesirable, that is to say, not permitted ('red').

Or: suppose now there are two customers, *custA* and *custB*. On Meyer's account, whenever *custA*'s account balance is negative, it is forbidden for *custB* to deposit or withdraw from his own account. This is surely wrong.

9.5 Relaxation of syntax

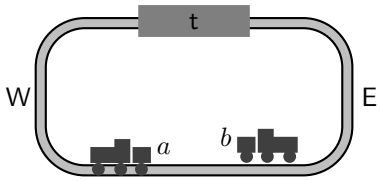
Permission laws of the form (15) and (16) are convenient but rather restrictive. For more flexibility, the $(\mathcal{C}+)^{++}$ language also allows distinguished fluent and action constants status and trans to be used explicitly in formulas and causal laws. The atoms $\mathsf{status} = \mathsf{red}$ and $\mathsf{trans} = \mathsf{red}$ can then be regarded as instances of what are sometimes called 'violation constants' in deontic logic. It is also easy to allow more 'shades' of red and green to allow different notions of permitted/legal/acceptable to be mixed. We will not employ that device in the examples discussed in this paper.

9.6 Example (trains)

The following example is used in (van der Hoek et al., 2004; Jamroga et al., 2004) to illustrate the use of alternating-time logic (ATL) (Alur et al., 2002) for determining the effectiveness of 'social laws' designed to co-ordinate the actions of agents in multi-agent system. The term is from (Shoham and Tennenholtz, 1992a,b, 1995; Moses and Tennenholtz, 1995) We will use the example for a different purpose, to illustrate how the permission component of $(\mathcal{C}+)^{++}$ can be used to analyse variants of the example in which agents may fail to obey social laws. More elaborate versions of the example, in which additional components are introduced in ordere to enforce compliance with the 'social laws', are discussed in (Sergot, 2005).

There are two trains, a and b, with a running clockwise round a double track, and b running anti-clockwise. There is a tunnel in which the double track becomes a single track. If the trains are both inside the tunnel at the same time they will collide. The tunnel can thus be seen as a kind of critical section, or as a resource for which the trains must compete.

There are obviously many ways in which the example can be formulated. The following will suffice for present purposes. Let simple fluent constants $loc(a)$ and $loc(b)$ represent the locations of trains a and b respectively. They both have possible values $\{\mathsf{W}, \mathsf{t}, \mathsf{E}\}$. For action constants, we take a and b with possible values $\{go, stay\}$. (Action constants $act(a)$ and $act(b)$ may be easier to read but we choose a and b for brevity.) Both are exogenous.



The $\mathcal{C}+$ action description representing the possible movements of the trains is as follows.

$$\mathsf{inertial}\ loc(a),\ loc(b) \tag{25}$$

train a moves clockwise:

$$\begin{array}{l} a = go\ \mathsf{causes}\ loc(a) = \mathsf{t}\ \mathsf{if}\ loc(a) = \mathsf{W} \\ a = go\ \mathsf{causes}\ loc(a) = \mathsf{E}\ \mathsf{if}\ loc(a) = \mathsf{t} \\ a = go\ \mathsf{causes}\ loc(a) = \mathsf{W}\ \mathsf{if}\ loc(a) = \mathsf{E} \end{array} \tag{26}$$

train b moves anti-clockwise:

$$\begin{array}{l} b = go\ \mathsf{causes}\ loc(b) = \mathsf{t}\ \mathsf{if}\ loc(b) = \mathsf{E} \\ b = go\ \mathsf{causes}\ loc(b) = \mathsf{W}\ \mathsf{if}\ loc(b) = \mathsf{t} \\ b = go\ \mathsf{causes}\ loc(b) = \mathsf{E}\ \mathsf{if}\ loc(b) = \mathsf{W} \end{array} \tag{27}$$

collisions:

$$\begin{array}{ll} collision\ \ \mathsf{iff}\ \ loc(a) = \mathsf{t} \wedge loc(b) = \mathsf{t} & \%\ \text{for convenience} \\ \mathsf{nonexecutable}\ a = go\ \mathsf{if}\ collision & \\ \mathsf{nonexecutable}\ b = go\ \mathsf{if}\ collision & \end{array} \tag{28}$$

collision is a statically determined Boolean fluent constant, introduced for convenience. Here and in the rest of this section, F iff G is used as shorthand for the pair of laws F if G and default $\neg F$.

Suppose, for the sake of an example, that we impose additional norms (social laws), as follows: no train is permitted to enter the tunnel unless the other train has just emerged. (We assume that this will be observed by the train that is preparing to enter. One of the variants of the example in (Sergot, 2005) introduces a communication mechanism–a semaphore–to allow the trains to communicate their positions to each other.) Will such a law be effective in avoiding collisions?

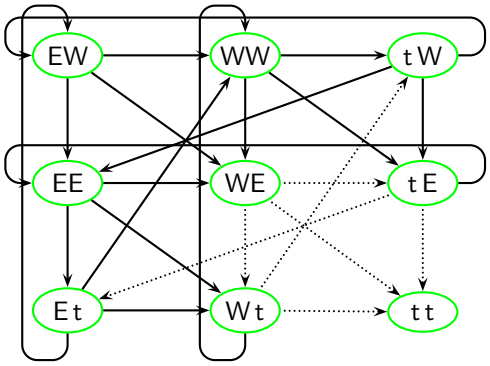


Figure 9.2: Coloured transition system for the trains example. A state label such as EW is short for $\{loc(a) = \mathsf{E}, loc(b) = \mathsf{W}\}$. Transition labels are omitted for clarity. Horizontal edges are transitions in which train a moves and b does not. Vertical edges are transitions in which train b moves and a does not. Diagonal edges are transitions in which both trains move. Reflexive edges, corresponding to transitions in which neither train moves, are omitted from the diagram to reduce clutter. Dotted lines indicate red transitions. All states and all other transitions, including the omitted reflexive ones, are green.

First, it is convenenient to define the following auxiliary action constants (all Boolean and non-exogenous):

$$\begin{array}{l} enter(a) \text{ iff } \ a = go \wedge loc(a) = \mathsf{W} \\ exit(a) \text{ iff } \ a = go \wedge loc(a) = \mathsf{t} \\ enter(b) \text{ iff } \ b = go \wedge loc(b) = \mathsf{E} \\ exit(b) \text{ iff } \ b = go \wedge loc(b) = \mathsf{t} \end{array} \tag{29}$$

Again, these are introduced merely for convenience; the example can be constructed easily enough without them. Now we formulate the social laws:

$$\begin{array}{l} \textsf{not-permitted } enter(a) \textsf{ if } loc(b) \neq \mathsf{W} \\ \textsf{not-permitted } enter(b) \textsf{ if } loc(a) \neq \mathsf{E} \end{array} \tag{30}$$

Let D_{trains} be the $(\mathcal{C}+)^{++}$ action description consisting of laws (25)–(30). The coloured transition system defined by D_{trains} is shown in Figure 9.2.

How do we test the effectiveness of the social laws (30)? Since the causal theory $\Gamma_1^{D_{\text{trains}}}$ encodes the transitions defined by D_{trains}, the following captures the property that if both trains comply with the social laws, no collisions will occur.

$$comp(\Gamma_1^{D_{\text{trains}}}) \models \neg collision[0] \wedge \mathsf{trans}[0] = \mathsf{green} \rightarrow \neg collision[1] \tag{31}$$

This can be checked, as in CCalc, by using a standard sat-solver to determine that the formula $comp(\Gamma_1^{D_{\text{trains}}}) \cup \{\neg collision[0] \wedge \mathsf{trans}[0] = \mathsf{green} \wedge collision[1]\}$ is not satisfiable. The property (31) is equivalently expressed as:

$$comp(\Gamma_1^{D_{\text{trains}}}) \models \neg collision[0] \wedge collision[1] \rightarrow \mathsf{trans}[0] = \mathsf{red} \tag{32}$$

which says that a collision occurs only following a transition in which either one train or both violate the norms.

Notice that $comp(\Gamma_1^{D_{\text{trains}}}) \not\models \mathsf{trans}[0] = \mathsf{green} \rightarrow \neg collision[1]$: as formulated by D_{trains}, the transition from a collision state to itself is green.

One major advantage of taking $\mathcal{C}+$ as the basic action formalism, compared to other action formalisms in AI, is its explicit transition system semantics. This enables a wide range of other analytical techniques to be applied. In particular, system properties can be expressed in the branching time temporal logic CTL and verified on the transition system defined by a $\mathcal{C}+$ or $(\mathcal{C}+)^{++}$ action description using standard model checking systems. (See e.g. (Clarke et al., 2000).)

We will say that a formula φ of CTL is *valid* on a (coloured) transition system $\langle S, \mathrm{I}(\sigma^{\mathrm{a}}), R, S_{\text{green}}, R_{\text{green}}\rangle$ defined by $(\mathcal{C}+)^{++}$ action description D when $s \cup \varepsilon \models \varphi$ for every $s \cup \varepsilon$ such that $(s, \varepsilon, s') \in R$ for some state s'. The definition is quite standard, except for a small adjustment to allow action constants in φ to continue to be evaluated on transition labels ε. (And we do not distinguish any particular set of initial states; all sets in S are initial states.) We will also say in that case that formula φ is valid on the action description D.

In CTL, the formula $\mathsf{AX}\,\varphi$ expresses that φ is satisfied in the next state in all future branching paths from now.[5] EX is the dual of AX: $\mathsf{EX}\,\varphi \equiv \neg\mathsf{AX}\,\neg\varphi$. $\mathsf{EX}\,\varphi$ expresses that φ is satisfied in the next state of some future branching path from now. The properties (31) and (32) can thus be expressed in CTL as follows:

$$\neg collision \wedge \mathsf{trans} = \mathsf{green} \rightarrow \mathsf{AX}\,\neg collision \tag{33}$$

or equivalently $\neg collision \wedge \mathsf{EX}\, collision \rightarrow \mathsf{trans} = \mathsf{red}$. It is easily verified by reference to Figure 9.2 that these formulas are valid on the action description D_{trains}. Also valid is the CTL formula $\mathsf{EX}\,\mathsf{trans} = \mathsf{green}$ which expresses that there is always a permitted action for both trains. This is true even in collision states, since the only available transition is then the one where both trains remain idle, and that transition is green. The CTL formula $\mathsf{EF}\, collision$ is also valid on D_{trains}, signifying that in every state there is at least one path from then on with *collision* true somewhere in the future.[6]

(Sergot, 2005) goes on to consider more elaborate versions of the example: in general, we want to be able to verify formally whether the introduction of additional control mechanisms—additional controller agents, communication devices, restrictions on agents' possible actions—are effective in ensuring that agents comply with the norms ('social laws') that govern their behaviour. These more elaborate versions can be analysed in similar fashion.

[5] $s_0 \cup \varepsilon_0 \models \mathsf{AX}\,\varphi$ if for every infinite path $s_0\,\varepsilon_0\,s_1\,\varepsilon_1\,\cdots$ we have that $s_1 \cup \varepsilon_1 \models \varphi$.

[6] $s_0 \cup \varepsilon_0 \models \mathsf{EF}\,\varphi$ if there is an (infinite) path $s_0\,\varepsilon_0\,\cdots\,s_m\,\varepsilon_m\,\cdots$ with $s_m \cup \varepsilon_m \models \varphi$ for some $m \geq 0$.

10 Conclusion

The language(s) $(\mathcal{C}+)^{++}$ provide an executable formalism for representing the institutional aspects of agent societies (specifically, the 'counts as' relation between actions) and a simple treatment of permission for expressing norms or 'social laws' that govern their behaviour.

Definite, atom-conjunctive action descriptions of $(\mathcal{C}+)^{+}$, which are the action descriptions of practical interest, can be implemented straightforwardly by translation to definite action descriptions of $\mathcal{C}+$. We thus have direct routes to implementation, using the established computational techniques for $\mathcal{C}+$, such as CCALC and translations to logic programs. And although in the paper we only illustrated the use of CTL and model checking techniques in connection with the permission component of $(\mathcal{C}+)^{++}$, it is clear that they can also be used to verify system properties of action descriptions of $(\mathcal{C}+)^{+}$, such as the auction protocol presented in Section 8.7.

The permission component of $(\mathcal{C}+)^{++}$ provides a means of distinguishing between permitted, acceptable, desirable ('green') states and transitions. It can be used as an extension of $\mathcal{C}+$, or in combination with the 'counts as' features, as an extension of $(\mathcal{C}+)^{+}$. Definite action descriptions of $(\mathcal{C}+)^{++}$ cannot be translated to $\mathcal{C}+$ because of the 'green-green-green constraint' but they can be translated to (definite) non-monotonic causal theories, requiring only very minor modification to existing computational tools such as CCALC. They can also be used with model checking systems, as illustrated briefly with the 'trains' example in Section 9.6. In addition to verifying system properties that hold if all agents/system components behave in accordance with norms/'social laws', we are able to analyse system properties that hold when agents/system components fail to comply with norms, and to analyse formally the effectiveness of introducing additional control, enforcement, and recovery mechanisms. This has been demonstrated on some small examples (see e.g. (Sergot, 2005)) but there is no reason to think that the techniques would not scale up to realistic examples, within the limits of existing model checking technology.

The semantical device employed in $(\mathcal{C}+)^{++}$, of partitioning states, and here also transitions, into 'ideal' ('green') and 'sub-ideal' ('red'), is familiar in the field of deontic logic. It is essentially the same device as is employed in 'Standard Deontic Logic' (SDL). It has very well known inadequacies, particularly in regard to the representation of 'contrary-to-duty' structures. To some extent these problems are mitigated in $(\mathcal{C}+)^{++}$ because it is a richer representational formalism than SDL and because it can deal with temporal aspects of 'contrary-to-duty' where these are present. However, there are benchmark problems in the deontic logic literature that will not receive adequate treatment in $(\mathcal{C}+)^{++}$. A detailed evaluation of the permission component of $(\mathcal{C}+)^{++}$, and a discussion of further possible refinements, is outside the scope of this paper and will be covered separately elsewhere.

The language(s) $(\mathcal{C}+)^{++}$ inherit from $\mathcal{C}+$ a number of very desirable features. Besides implementation methods, and support for concurrent actions, nondeterminism, and indirect effects of actions, the underlying formalism of causal theories provides a general purpose non-monotonic framework for expressing defeasible general rules and defeasible effects of actions.

The language(s) $(\mathcal{C}+)^{++}$ also inherit a number of important limitations. One limitation is that causal rules can refer only to states and transitions at most one time-step away from each other. This makes it impossible to refer directly to the values of fluents further in the past, or to features of earlier transitions. In the auction example of Section 8.7, for example, it was necessary to introduce additional fluents to represent the last bid made in order to encode a simple revocation mechanism. More complicated examples soon become unwieldy. We might wish to say, for example, that three attempted invalid bids by an agent in an auction are permitted, but a fourth is not. Relying on additional fluents to record the relevant fragment of history not only increases the number of states (exponentially) but also tends to make the representation obscure. It is also impossible to say, directly, that the execution of one action causes the execution of another action at the next time-step, or at some other time in the future. These are limitations of $\mathcal{C}+$ rather than of the causal theories into which $\mathcal{C}+$ is translated: Craven and Sergot (2005) present a generalisation of $\mathcal{C}+$ which removes this restriction, though it provides only a partial solution and the corresponding $(\mathcal{C}+)^{++}$ extensions remain to be developed.

Other limitations of $\mathcal{C}+$ and $(\mathcal{C}+)^{++}$ are limitations of transition systems generally. States and transitions are global, and so lack the structure required to model large numbers of interacting components, such as multi-agent systems. Transitions have no (explicit) duration, and there is no support for time metrics: encoding delays and deadlines in $\mathcal{C}+$ is awkward at best. Naturally there are other structures in computer science that have been designed to overcome the limitations of transition systems. One can imagine devising similar representational formalisms based on these other structures. The details are far from trivial of course.

Besides addressing these various issues, current development of $(\mathcal{C}+)^{++}$ includes the following topics:

- technical investigation of logical properties of $\mathcal{C}+$ and of $(\mathcal{C}+)^{++}$ (see (Sergot and Craven, 2005) for a preliminary study);
- experiments with the use of the CCALC-based implementation of $(\mathcal{C}+)^{++}$ as the reasoning engine for the 'society simulator' developed by Alexander Artikis (Artikis et al., 2002) in place of the event calculus and $\mathcal{C}+$ currently employed;
- refinement of the representational formalism to provide more structure in states, in two respects:
  - support for multiple institutions (multiple 'counts as' relations)
  - the introduction of local states, and local actions, for agents (and environment) for modelling multi-agent systems;
- development of the $\mathcal{EC}+$ implementation described in (Craven and Sergot, 2003) which provides an event calculus style of computation for queries on (a subset of) $\mathcal{C}+$ action descriptions;
- investigations of the use of $\mathcal{C}+$ and $(\mathcal{C}+)^{++}$ as input languages for model checking systems, specifically NuSMV (Cimatti et al., 2002).

Acknowledgements

A preliminary version of $(\mathcal{C}+)^{++}$ was produced in the course of the EU funded FET project AlfeBiite (IST-1999-10298). The author wishes to express his thanks to Alex Artikis and Rob Craven for many valuable discussions and suggestions, to Joohyung Lee for helpful clarifications of features of the $\mathcal{C}+$ language, and to Jeremy Pitt for his patience.

References

V. Akman, S. T. Erdoğan, J. Lee, V. Lifschitz, and H. Turner. Representing the Zoo World and the Traffic World in the language of the Causal Calculator. *Artificial Intelligence*, 153:105–140, 2004.

R. Alur, T.A. Henziger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, September 2002.

A. Artikis, J. Pitt, and M. J. Sergot. Animated specification of computational societies. In Cristiano Castelfranchi and W. Lewis Johnson, editors, *Proc. 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02), Bologna*, pages 1053–1062. ACM Press, July 2002.

A. Artikis, M. J. Sergot, and J. Pitt. An executable specification of an argumentation protocol. In *Proc. 9th International Conference on Artificial Intelligence and Law (ICAIL'03), Edinburgh*, pages 1–11. ACM Press, 2003a.

A. Artikis, M. J. Sergot, and J. Pitt. Specifying electronic societies with the Causal Calculator. In Fausto Giunchiglia, James Odell, and Gerhard Weiss, editors, *Agent-Oriented Software Engineering III. Proc. 3rd International Workshop (AOSE 2002), Bologna, July 2002*, LNCS 2585, pages 1–15. Springer, 2003b.

José Carmo and Andrew J. I. Jones. Deontic database constraints, violation and recovery. *Studia Logica*, 57(1):139–165, 1996.

A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking. In *Proc. International Conference on Computer-Aided Verification (CAV 2002), Copenhagen, July 2002*, LNCS 2404. Springer, 2002. See `http://nusmv.irst.itc.it`.

E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, Cambridge, 2000.

R. A. Craven and M. J. Sergot. $\mathcal{EC}+$—efficient computation of queries for the language $\mathcal{C}+$. Technical report, Department of Computing, Imperial College, London, UK, 2003.

Robert Craven and Marek Sergot. Distant causation in $\mathcal{C}+$. *Studia Logica*, 79 (1):73–96, February 2005.

Thomas Eiter, Wolfgang Faber, Nicola Leone, Gerald Pfeifer, and Axel Polleres. System description: The DLV-K planning system. In Thomas Eiter, Wolfgang Faber, and Mirosław Truszczyński, editors, *Proc. 6th International Conference on Logic Programming and Nonmonotonic Reasoning*, LNAI 2173, pages 413–416. Springer-Verlag, 2001.

J. Gelati, G. Governatori, A. Rotolo, and G. Sartor. Declarative power, repre-

sentation, and mandate. A formal analysis. In *Proceedings of Conference on Legal Knowledge and Information Systems (JURIX)*, 2002.

Jonathan Gelati, Antonino Rotolo, Giovanni Sartor, and Guido Governatori. Normative autonomy and normative co-ordination: Declarative power, representation and mandate. *Artificial Intelligence and Law*, 12:53–81, 2004.

Michael Gelfond and Vladimir Lifschitz. Action Languages. *Electronic Transactions on AI*, 3(16), 1998. `http://www.ep.lin.se/ea/cis/1998/016`.

Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain, and Hudson Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153: 49–104, 2004.

Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, and Hudson Turner. Causal laws and multi-valued fluents. In *Proc. of the Fourth Workshop on Nonmonotonic Reasoning, Action, and Change, Seattle*, August 2001.

Enrico Giunchiglia and Vladimir Lifschitz. An action language based on causal explanation: Preliminary report. In *Proc. AAAI-98*, pages 623–630. AAAI Press, 1998.

Enrico Giunchiglia and Vladimir Lifschitz. Action languages, temporal action logics, and the situation calculus. In *Working Notes of the IJCAI-99 Workshop on Nonmonotonic Reasoning, Action, and Change*, 1999.

A. I. Goldman. *A Theory of Human Action*. Prentice-Hall, New Jersey, 1970.

W. Jamroga, W. van der Hoek, and M. Wooldridge. On obligations and abilities. In Alessio Lomuscio and Donald Nute, editors, *Proc. 7th International Workshop on Deontic Logic in Computer Science (DEON'04), Madeira, May 2004*, LNAI 3065, pages 165–181. Springer, 2004.

A. J. I. Jones and M. J. Sergot. A formal characterisation of institutionalised power. *Journal of the IGPL*, 4(3):429–445, 1996. Reprinted as (Jones and Sergot, 1997).

A. J. I. Jones and M. J. Sergot. A formal characterisation of institutionalised power. In E. G. Valdés, W. Krawietz, G. H. von Wright, and R. Zimmerling, editors, *Normative Systems in Legal and Moral Theory. Festschrift for Carlos E. Alchourrón and Eugenio Bulygin*, pages 349–367. Duncker & Humboldt, Berlin, 1997.

Vladimir Lifschitz. On the logic of causal explanation. *Artificial Intelligence*, 96:451–465, 1997.

Vladimir Lifschitz and Hudson Turner. Representing transition systems by logic programs. In *Proc. Fifth International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 92–106, 1999.

A. Lomuscio and M. J. Sergot. Deontic interpreted systems. *Studia Logica*, 75 (1):63–92, October 2003.

Norman McCain and Hudson Turner. Causal theories of action and change. In *Proc. AAAI-97, Providence, RI*, pages 460–465. AAAI Press, 1997.

R. van der Meyden. The dynamic logic of permission. *Journal of Logic and Computation*, 6(3):465–479, 1996.

John-Jules Ch. Meyer. A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic*, 29(1):109–136, 1988.

Y. Moses and M. Tennenholtz. Artificial social systems. *Computers and Artificial Intelligence*, 14(6):533–562, 1995.

Ilkka Niemelä. Logic programs with stable model semantics as a constraint logic programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25:241–273, 1999.

H. Prakken. Formalising Robert's rules of order. Technical Report 12, GMD – German National Research Center for Information Technology, 1998.

H. Prakken and T. Gordon. Rules of order for electronic group decision making – a formalization methodology. In J. Padget, editor, *Collaboration between Human and Artificial Societies*, LNCS 1624, pages 246–263. Springer, 1999.

John R. Searle. *Speech Acts*. Cambridge University Press, 1969.

M. J. Sergot. Modelling unreliable and untrustworthy agent behaviour. In B. Dunin-Keplicz, A. Jankowski, A. Skowron, and M. Szczuka, editors, *Monitoring, Security, and Rescue Techniques in Multiagent Systems*, Advances in Soft Computing, pages 161–178. Springer-Verlag, 2005.

Marek Sergot and Robert Craven. Some logical properties of nonmonotonic causal theories. In *Proc. Eighth International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR'05), Calabria, September 2005*, LNAI 3662. Springer, 2005. In press.

Y. Shoham and M. Tennenholtz. Emergent conventions in multi-agent systems. In *Proceedings of Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 225–231, 1992a.

Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proceedings of Tenth National Conference on Artificial Intelligence (AAAI'92), San Diego*, pages 276–281. The AAAI Press/ The MIT Press, 1992b.

Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, 73(1-2):231–252, 1995.

Hudson Turner. A logic of universal causation. *Artificial Intelligence*, 113: 87–123, 1999.

W. van der Hoek, M. Roberts, and M. Wooldridge. Social laws in alternating time: Effectiveness, feasibility, and synthesis. Technical report, Dept. of Computer Science, University of Liverpool, 2004. Submitted.