Agents in Proactive Environments

Dov Gabbay
Department of Computing
Imperial College
London
England

Rolf Nossum [1]
Department of Mathematics
Agder College
Kristiansand
Norway

Michael Thielscher [1]
FG Intellektik
University of Technology
Darmstadt
Germany

authors' email: { dg, rn7, mt11} @doc.ic.ac.uk
phone: +44 171 594 8205
fax: +44 171 594 8201

Abstract. Agents situated in proactive environments are acting autonomously while the environment is evolving alongside, whether or not the agents carry out any particular actions. A formal framework for simulating and reasoning about this generalized kind of dynamic systems is proposed. The capabilities of the agents are modeled by a set of conditional rules in a temporal-logical format. The environment itself is modeled by an independent transition relation on the state space. The temporal language is given a declarative semantics on the basis of an abstract, general model structure for formal specifications of proactive environments.

1 Introduction

To motivate what follows, we discuss some aspects of scenarios involving agents and their environment, both capable of changing the state of affairs of the world. The environment in these scenarios is evolving freely, whether or not there is any action on the part of the agent. This seems realistic, many real-world scenarios are like that. This contrasts with the scenario classes that have mainly been studied, where the environment is perceived as a purely reactive mechanism; e.g. in [21] pp.16-17 we find: "The definitions in this book will be made in such a way that if no action is invoked by the ego, then the world will advance by one single timestep while keeping all feature-values unchanged." Scenarios like the following go beyond this idealistic view of the world pausing whenever an agent decides not to act:

[1] Most of the work was carried out while these authors were visiting Imperial College. An extended abstract of this work has been presented at KI'97. Some of the results have been presented previously at IJCAI'95.

- An interest-bearing bank deposit accrues interest from time to time, increasing the balance of the account. The owner of the account can meanwhile make deposits and withdrawals, not necessarily synchronized with the times at which interest is added. The rate of interest on such accounts varies with the bank, with the size of the deposit, with inflation, trading balance and other economic parameters in that particular country, and ultimately on the current state of the world economy.

- Users of a timesharing computer compete for unique access to certain system resources. At irregular times, the timesharing system itself reserves access to these resources, e.g. for maintenance.

  A free resource can be claimed by a user by performing a certain system call. The resource then becomes busy until that user performs a releasing system call.

  One day, two users both require simultaneous unique access to the same pair of resources, and each of them gets hold of one resource by performing the appropriate system call. They are then deadlocked, and a conflict resolution tactic must be deployed, either by the users themselves or the system environment.

These examples are intended to convey the flavour of agents in an environment which is changing in a way that is not purely reactive to individual actions, and which is too complex to be described fully as a multi-agent system. We propose a formal framework for modeling dynamic environments with situated agents whose actions influence the development in the course of time. We start with a brief discussion on and inert vs. transient state components (Section 2). In Section 3, we introduce a temporal execution language named TEAL. It allows for the specification of both proactive environments and the effects on it caused by the performance of actions. An example domain, viz. competition for access to system resources as mentioned above, is formalized by means of TEAL in Section 4. Section 5 contains a brief discussion on the operational semantics followed by an example execution scenario (Section 6) which describes a particular evolution of the dynamic system formalized in Section 4. In Section 7, we develop a general, abstract model structure for specifications of dynamic systems with proactive environment. On this basis, in Section 8 we provide a declarative semantics for TEAL. In Section 9 we extend the declarative semantics to both nondeterminism and conflict resolution. The paper ends with a discussion of related work in Section 10, and some concluding remarks in Section 11.

2 Inertia vs. transience

We take the state of affairs in the world to consist of a countable set F of atoms (called *fluents* [18]), some of which are designated as *inert*. These are the fluents that are thought of as stable if not explicitly changed. The agent, as well as the environment, can change them. For example, the lights in a room are inert; when the switch is flicked, the status of the light changes, and remains inert until the bulb burns out, or the switch is flicked again. The lights in the stairways of some buildings are fitted with a time-delay circuit,

which turns them off when they have been on for a while; this can be viewed as the environment acting on the fluent, independently of the agent that turned it on.

The non-inert fluents are *transient*,[2] i.e. lasting for a single time unit. This is an approximation to the transient nature of events like the sound of a doorbell, or a flash of lightning.

Some transient fluents are designated as *actions*, and are carried out by agents or their environment. The flicking of a switch is an action, and therefore transient, but the status of the light, which is an effect of the flicking, is inert. Ringing a doorbell is an action, and so is setting off a lightning flash in a thunderstorm, but the latter is an action available only to the environment.

A flash of lightning can have drastic effects, which can greatly influence the further evolution of agent scenarios, for instance by rendering some courses of action impossible for some agents.

Flashes of lightning are in general not provoked by any action on the part of an agent, so it would be unnatural to have a model where the environment was restricted to responding directly to actions of an agent. On the other hand, the agent can influence the evolution of the environment, for instance by erecting a lightning rod.

3 TEAL—A Temporal Executable Action Language

We proceed to give a formula language similar to that in [5], with an informal interpretation of some formulas as describing the actions of an agent. A precise formal semantical interpretation in terms of a logic of proactive environments follows in a later section.

The language elements are the following. Propositional fluents, denoted p, q, r, etc; classical connectives with their usual semantics, as well as temporal modalities $\bigcirc$ ("tomorrow"), $\bullet$("yesterday"); and actions, which are denoted by atomic terms like *shoot*, and ground terms like $make\_deposit(250)$.

There is a distinguished predicate $exec(...)$ on actions, generating atomic propositions from action terms.

3.1 Action Rules

Fluents and negated fluents are called literals. Literals are combined with connectives and modalities into formulas. Literals with 1 or more $\bullet$on them are called past literals, and literals with 1 or more $\bigcirc$s on them are called future literals.

Clauses of the form

$$past\ literals \wedge exec(a) \rightarrow future\ literals$$

are called action rules.

Action rules are interpreted by evaluating the past literals, and if they are true, recording the action as having been carried out at the present time, and performing the future

[2] called *momentary* in [14]

literals at the appropriate points in time. More precisely, a rule

$$\bigwedge_i \bullet^i e_i \wedge exec(a) \rightarrow \bigwedge_j \bigcirc^j e_j$$

is eligible at time t if $t \models \bigwedge_i \bullet^i e_i$, cfr. the operational semantics below. Whether it is actually carried out depends on constraints, and on possible conflicts with other rules.

3.2 Constraints

It follows from a separation result by Gabbay [6] that a large class of temporal formulas can be normalized as action rules. Formulas outside this class can be interpreted as constraints on the execution of action rules.

A constraint ψ prevents a rule $\bigwedge_i \bullet^i e_i \wedge exec(a) \rightarrow \bigwedge_j \bigcirc^j e_j$ from execution at time t if $t \models \psi \rightarrow \neg exec(a)$, and also conflicts with it if $t + j \models \psi \rightarrow \neg e_j$ for some j.

4 Example: timesharing and deadlock

We take a closer look at one of the examples mentioned in the introduction, namely the competing users of a timesharing computer system.

There are two users, competing for access to both of two distinct resources. Each resource can only be accessed by one user at a time. Each user wants control over both resources simultaneously in order to accomplish a certain task. Also, the operating system may reserve and release these resources, say for periodic maintenance, on a schedule that is left unspecified here.

We set out some action terms and fluents which will help formalize this picture. A resource r, which is not in use, can be reserved for a user u by an action *takes*. Therefore, for each user u and each resource r we have action terms

$$u \; takes \; r$$

A user u, who has acquired both resources, accomplishes the task at hand by performing the action

$$u \; succeeds$$

and then releases the resources.

Finally, we need fluents

$$u \; has \; r$$

for each user u and resource r, to describe who, if anyone, has reserved access to each resource.

Supposing the two competing users are u_1 and u_2, the operating system is *os*, and the contended resources are r_1 and r_2, we can describe the race to reserve both resources by the following TEAL clauses:

$$(T1) \quad \neg u_1\ has\ r_1 \wedge \neg u_2\ has\ r_1 \wedge \neg os\ has\ r_1 \wedge exec(u_1\ takes\ r_1) \rightarrow \bigcirc u_1\ has\ r_1$$
$$(T2) \quad \neg u_1\ has\ r_2 \wedge \neg u_2\ has\ r_2 \wedge \neg os\ has\ r_2 \wedge exec(u_1\ takes\ r_2) \rightarrow \bigcirc u_1\ has\ r_2$$
$$(T3) \quad \neg u_1\ has\ r_1 \wedge \neg u_2\ has\ r_1 \wedge \neg os\ has\ r_1 \wedge exec(u_2\ takes\ r_1) \rightarrow \bigcirc u_2\ has\ r_1$$
$$(T4) \quad \neg u_1\ has\ r_2 \wedge \neg u_2\ has\ r_2 \wedge \neg os\ has\ r_2 \wedge exec(u_2\ takes\ r_2) \rightarrow \bigcirc u_2\ has\ r_2$$

If and when a user succeeds in getting both resources, then the task at hand is performed, and both resources released:

$$(T5) \quad u_1\ has\ r_1 \wedge u_1\ has\ r_2 \wedge exec(u_1\ succeeds) \rightarrow \bigcirc \neg u_1\ has\ r_1 \wedge \bigcirc \neg u_1\ has\ r_2$$
$$(T6) \quad u_2\ has\ r_1 \wedge u_2\ has\ r_2 \wedge exec(u_2\ succeeds) \rightarrow \bigcirc \neg u_2\ has\ r_1 \wedge \bigcirc \neg u_2\ has\ r_2$$

The operating system can intervene at any time by reserving any of the resources for its own purposes, in unspecified ways. We can imagine that there exist TEAL clauses describing this, but that they are unavailable for analysis.

The following constraints describe the exclusiveness of access to the resources:

$$(C1) \quad u_1\ has\ r_1 \rightarrow \neg u_2\ has\ r_1 \wedge \neg os\ has\ r_1$$
$$(C2) \quad u_1\ has\ r_2 \rightarrow \neg u_2\ has\ r_2 \wedge \neg os\ has\ r_2$$
$$(C3) \quad u_2\ has\ r_1 \rightarrow \neg u_1\ has\ r_1 \wedge \neg os\ has\ r_1$$
$$(C4) \quad u_2\ has\ r_2 \rightarrow \neg u_1\ has\ r_2 \wedge \neg os\ has\ r_2$$
$$(C5) \quad os\ has\ r_1 \rightarrow \neg u_1\ has\ r_1 \wedge \neg u_2\ has\ r_1$$
$$(C6) \quad os\ has\ r_2 \rightarrow \neg u_1\ has\ r_2 \wedge \neg u_2\ has\ r_2$$

All of the above clauses taken together, including the existing but unknown clauses for the operating system, form a specification of ways in which the competition for resources may evolve. There are a multitude of different TEAL executions of them, corresponding to nondeterministic choice of clauses where the preconditions are fulfilled, and conflict resolution between clauses where the effects are contradictory or in violation of the constraints.

Some of the possible executions result in undesired states, where

$$(DL) \quad u_1\ has\ r_1 \wedge u_2\ has\ r_2 \vee u_1\ has\ r_2 \wedge u_2\ has\ r_1$$

It is tempting to call this eventuality deadlock, since in every continuation of such states

$$\neg exec(u_1\ succeeds) \wedge \neg exec(u_2\ succeeds)$$

holds. It is however more properly conceived of as a form of livelock, since other computations may well proceed normally, e.g. the activities of the operating system.

Deadlock avoidance is a matter of conflict resolution, and in this example it is natural to include the negation of DL as another constraint:

$$(C7) \quad (u_1\ has\ r_1 \rightarrow \neg u_2\ has\ r_2) \ \wedge \ (u_1\ has\ r_2 \rightarrow \neg u_2\ has\ r_1)$$

5 Operational semantics

For simplicity we take integer time with finite past. Action rules can be executed at points of time, resulting in a change in the future state.

For the moment, we disregard possible conflicts between different rules, and concentrate on the operational semantics of executing a single rule at a certain timepoint t. Issues of competing clauses, and in the case of concurrency, conflicting ones, are dealt with elsewhere.

We rely on an underlying notion of truth of fluents at timepoints. Intuitively, we may visualize the development of the state through time as a two-dimensional matrix, indexed in one direction by fluents and in the other by time. For finite-past integer time, the matrix will be infinite in one direction, that of advancing time. For the sake of uniformity, we assume that the past-most timepoint is 1.

Thus, operationally speaking, an execution model is a set of fluents together with a mapping $\models$ of timepoints and fluents to truth values. For a fluent f and a timepoint t, the realtion $t \models f$ is either true or false. This matrix can be maintained as a temporal database, using whatever database technology one prefers, e.g. that of [17]. To check whether $t \models \varphi$ holds, one can query the database for, say, $?t : \varphi$. Formulas composed by propositional connectives or temporal modalities must satisfy the usual inductive definitions:

- $t \models \neg\varphi$ iff not $t \models \varphi$
- $t \models \psi \rightarrow \varphi$ iff $t \models \psi$ implies $t \models \varphi$
- $t \models \blacklozenge$ iff $t - 1 \models \varphi$
- $t \models \bigcirc\varphi$ iff $t + 1 \models \varphi$

To execute a rule at time t, the preconditions are queried from the temporal database. If they are met, the action fluent $exec(a)$ is recorded as true at time t, and the effect fluents are entered into the database at the appropriate times as indicated by their temporal prefixes. Database management rules and constraints will need to be accounted for.

There is no guarantee that these updates of the temporal database can be made consistently. Conflict resolution between rules that have inconsistent effects, is dealt with in Section 9.

The definition of TEAL does not address any technicalities of backtracking or forking required to implement non-determinism. This is left to the implementor, who will call for one or the other implementation technique, depending mainly on the number of independent processing modules that are available.

6 Execution scenarios

Returning to clauses $(T1) - (T6)$, which describe the two competing users of a computer system, we recall that some possible action sequences result in deadlock, but not all.

Every particular possible course of events can be retrieved after the fact by projecting the resulting temporal database onto the *exec*(...) action fluents, revealing which actions were done when.

A happy course of events would be this: at time 1, u_1 took hold of resource r_1, then at time 2, user u_1 took hold of r_2 also, enabling u_1's success and release of the resources at time 3. Then u_2 did a similar sequence of actions from time 4 onwards, while all the time the environment, the operating system, did nothing at all to restrict access the resources. We can model this scenario by a sequence of sets of action fluents, indexed by timepoints from 1 onwards:

$$\begin{array}{ll}[& \{exec(u_1\ takes\ r_1)\}, \{exec(u_1\ takes\ r_2)\}, \{exec(u_1\ succeeds)\}, \\ & \{exec(u_2\ takes\ r_1)\}, \{exec(u_2\ takes\ r_2)\}, \{exec(u_2\ succeeds)\}\ ]\end{array}$$

To complicate matters a little bit, without running into deadlock, suppose that the environment takes out resource r_2 for maintenance in the time interval $4 \ldots 12$. This makes u_2 have to wait a while, but eventually both users succeed. To formalize it, we need to specify action patterns of the operating system. Let user *os* have available the action *os takes r* for $r = r_1, r_2$ in similar ways as in $(T1) - (T4)$, and let there be an action *drops* as follows:

$$(DR) \quad u\ has\ r \wedge exec(u\ drops\ r) \rightarrow \bigcirc \neg u\ has\ r$$

for arbitrary u and r. Then the following execution scenario leads to eventual success for both of u_1 and u_2:

$$\begin{array}{ll}[& \{exec(u_1\ takes\ r_1)\}, \{exec(u_1\ takes\ r_2)\}, \{exec(u_1\ succeeds)\}, \\ & \{exec(u_2\ takes\ r_1), exec(os\ takes\ r2)\}, \{\}, \{\}, \{\}, \{\}, \{\}, \{\}, \{\}, \\ & \{exec(os\ drops\ r2)\}, \{exec(u_2\ takes\ r_2)\}, \{exec(u_2\ succeeds)\}\ ]\end{array}$$

Operationally speaking, an execution model wrt. an execution scenario $\sigma = [a_1, \ldots, a_n]$ obtains as follows. A truth value assignment F_0 is updated wrt. the *exec* instances in a_1, thus yielding the initial state of the database at time $t = 1$. The database then evolves according to the underlying action rules but with constant updating of *exec* instances as specified in σ until time $t = n$. After completing the execution scenario, the database may continue evolving on its own, i.e., without further exogenous actions.

Execution models wrt. execution scenarios correspond directly to formal developments in the logic of proactive environments (see below), and this will provide us with a declarative semantics for reasoning about TEAL specifications.

7 The Logic of Proactive Environments

We proceed by developing a logic-oriented, formal theory of dynamically changing worlds and the process of acting in them. Our model-theoretic approach comes with two main features. First, the notion of concurrency is intrinsic to the theory. Second, we can easily model delayed effects of actions by initiating additional independent events which eventually trigger a particular effect.

When specifying a dynamic system, a major challenge is to find a compact description of the underlying causal model, which defines the space of possible state changes in the course of time. Our theory includes these two fundamental concepts: First, the *persistence assumption*,[3] which enables one to state explicitly only what changes during a single state transition while everything else is implicitly assumed to remain unchanged. Second, *atomic causal laws* are used to state relationships between single cause-effect-pairs. Since usually several of these atomic laws apply to the current state of a dynamic system, a combination of such laws determines the entire transition step. The use of atomic causal relationships is especially helpful in theories which involve concurrency of actions and events.

The major modes of reasoning with formal specifications of dynamic systems are the following:

- In *temporal projection*, one is interested in the result of performing a particular sequence of actions starting in a particular state of the system.
- In *planning*, the question is whether a sequence of actions, a *plan*, can be found whose execution results in a system state that satisfies a given goal.
- In *postdiction*,[4] one is faced with a narrative which is represented by a number of observations regarding a system's development during a specific period. These observations are used to derive more information about what has happened.

7.1 Specifying Dynamic Systems

A formal specification of a dynamic system has two components. First, we need a collection of fluents to describe particular states of the system. Second, the behavior of the system as regards state transitions needs to be determined. Let us, for the moment, focus on deterministic dynamic systems; nondeterminism is dealt with in Section 9. This restriction is reflected in the following definition, where a transition function maps a state into a unique successor state.

Definition 7.1 A *deterministic, propositional dynamic system* is a pair $(\mathcal{F}, \Phi)$ consisting of a finite set of symbols $\mathcal{F}$, called *fluents*, and a partially defined mapping $\Phi : \mathcal{C} \mapsto \mathcal{C}$, called *causal model*. The range of the latter is a particular set of subsets of $\mathcal{F}$, i.e., $\mathcal{C} \subseteq 2^{\mathcal{F}}$.

Each subset s of $\mathcal{F}$ determines a (not necessarily possible) *state* of the dynamic system at hand. Each fluent $f \in s$ is then said to be *true* in s while each fluent $f \in \mathcal{F} \setminus s$ is taken to be *false*. The set $\mathcal{C}$ is intended to contain all so-called *consistent* states—only for these states s the *successor* state $\Phi(s)$ is defined by the causal model.

Based on the notion of truth of fluents in states, we can construct (propositional) formulae and extend the notion of truth as usual:

[3] also called *frame assumption* or *inertia principle*

[4] called *chronicle completion* in [21]

Definition 7.2 Let $\mathcal{F}$ be a set of fluents. The set of *fluent formulae* (based on $\mathcal{F}$) is the smallest set such that each element $f \in \mathcal{F}$ is a fluent formula; and if F and G both are fluent formulae then so are $\neg F$, $(F \wedge G)$, $(F \vee G)$, and $(F \rightarrow G)$.

Given a state $s \subseteq \mathcal{F}$ and a fluent formula F, the notion of F being *true in* s, written $s \models F$, is inductively defined as follows:

- $s \models f$ iff $f \in s$, where $f \in \mathcal{F}$.
- $s \models \neg F$ iff $s \not\models F$.
- $s \models (F \wedge G)$ iff both $s \models F$ and $s \models G$.
- $s \models (F \vee G)$ iff $s \models F$ or $s \models G$ (or both).
- $s \models (F \rightarrow G)$ iff $s \not\models F$ or $s \models G$ (or both).

Fluent formulae can be used, for instance, to specify consistency of states more compactly by means of a particular formula C such that a state $s \in \mathcal{F}$ is defined to be consistent iff $s \models C$. Then the set $\mathcal{C}$, which contains all consistent states of a dynamic system (see Definition 7.1), is implicitly given by $\mathcal{C} = \{s \subseteq \mathcal{F} \mid s \models C\}$. Some distinguished fluents $\mathcal{F}_a \subset \mathcal{F}$ are called *actions*; these are fluents which an agent can force to become true in the current state in order to influence the system's behavior.

Example 1 The *Yale Shooting domain* [12], in several variants, shall be used as the running example of this section. To formalize a preliminary version, consider the set of fluents $\mathcal{F} = \{alive, loaded, load, shoot\}$—where *alive* and *loaded* are used to describe the state of the turkey and the gun, respectively, while *load* and *shoot* are action fluents representing the events of loading the gun and shooting, respectively. Then in the state $s = \{alive, load\} \subseteq \mathcal{F}$, for instance, the turkey is alive, the gun is unloaded, and some agent intends to perform a *load* action. The successor state $\Phi(s)$ might then be defined as $\{alive, loaded\}$, indicating that the turkey is still alive and that the gun is now loaded. Furthermore, one might wish to specify that the gun cannot simultaneously be loaded and shot with. This can be achieved by means of the constraint $C = \neg(load \wedge shoot)$. We then have, say, $s \models C$ on account of $\{alive, load\} \not\models shoot$. ∎

7.2 Causal Laws

The main challenge when specifying a dynamic system is to find a compact representation of the corresponding causal model Φ. The most fundamental concept related to this is the principle of *persistence*, which enables one to only specify the changes during a particular state transition; all other fluents are implicitly assumed to keep their value.

As motivated in Section 2, we make a distinction between so-called inert fluents $\mathcal{F}_i$ that "tend to persist," i.e., which are assumed to keep their value until the contrary is explicitly stated (and, hence, to which the persistence assumption should apply), and *transient* fluents $\mathcal{F}_t$ that "tend to disappear." As an important subclass of transient fluents we have the action fluents $\mathcal{F}_a$. Altogether, the set of fluents $\mathcal{F}$ describing a dynamic system consists essentially of three components $(\mathcal{F}_i, \mathcal{F}_t, \mathcal{F}_a)$ where $\mathcal{F}_a \subseteq \mathcal{F}_t$ and $\mathcal{F}_i \cap \mathcal{F}_t = \{\}$.

Based on this sophistication, the persistence principle is integrated into our framework by defining that, for each state s, the successor state $\Phi(s)$ is specified via an associated triple of sets of fluents $\langle i^-, i^+, t^+\rangle$. Here, i^- contains the inert fluents which change their truth value to false during the state transition, hence which are removed from s; i^+ contains the inert fluents which change their truth value to true, hence which are added to s; and t^+ contains all transient fluents which are true in $\Phi(s)$. All other inert fluents in s continue to be (and no other inert fluents become) element of $\Phi(s)$ while all transient fluents except for those in t^+ shall not be contained in the resulting state.

Example 2 Consider an extension of the Yale Shooting domain with the inert fluents $\mathcal{F}_i = \{alive, loaded\}$ and the transient fluents $\mathcal{F}_t = \{bang, bullet, load, shoot\}$, where the additional fluents *bang* and *bullet* describe, respectively, the temporary acoustical occurrence of a shot and a flying bullet. We then might specify the following:

$$s = \{alive, loaded, shoot\} \ : \ \langle i^-, i^+, t^+\rangle = \langle\{loaded\}, \{\}, \{bang, bullet\}\rangle \quad (1)$$

$$s = \{alive, bang, bullet\} \ : \ \langle i^-, i^+, t^+\rangle = \langle\{alive\}, \{\}, \{\}\rangle \quad (2)$$

Put in words, shooting with a previously loaded gun causes the gun to become unloaded ($loaded \in i^-_{(1)}$) and the occurrence of two events, *bang* and *bullet*; a flying bullet hits the turkey and, hence, causes it to drop dead ($alive \in i^-_{(2)}$). This example illustrates how our paradigm allows for a natural formalization of delayed effects (here, the turkey being shot as the final result of having shot with the gun). With this specification we obtain, for instance, $\Phi(\Phi(\{alive, loaded, shoot\})) = \Phi(\{alive, bang, bullet\}) = \{\}$. Notice that finally both *bang* and *bullet* disappear automatically because they are transient. ∎

The persistence assumption allows for a more compact, implicit definition of successor states $\Phi(s)$. Yet the above formalization still requires an exhaustive description as regards the space of states. Therefore, the second major principle of specifying the behavior of a dynamic system consists in splitting the definition of a single state transition into separate *atomic* laws of causality, which then apply to a variety of different states. This is particularly crucial in theories which involve concurrency since it enables one to specify the effects of each single action (or event like *bullet*) separately:

Definition 7.3 Let $\mathcal{F} = \mathcal{F}_i \dot\cup \mathcal{F}_t$ be a set of inert and transient fluents. A structure $c : \langle i^-, i^+, t^+\rangle$ is called a *causal law* if $c \subseteq \mathcal{F}$, called the *condition*; $i^-, i^+ \subseteq \mathcal{F}_i$; and $t^+ \subseteq \mathcal{F}_t$.

A causal law is applicable in a state whenever its condition is contained in the state description.[5] For notational convenience, to have access to the four components of some causal law $\ell = c : \langle i^-, i^+, t^+\rangle$ we introduce the four functions $cond(\ell) \stackrel{\text{def}}{=} c$, $inert^-(\ell) \stackrel{\text{def}}{=} i^-$, $inert^+(\ell) \stackrel{\text{def}}{=} i^+$ and $transient^+(\ell) \stackrel{\text{def}}{=} t^+$. We furthermore use the following abbreviation to

[5]It is for the sake of simplicity that we restrict the condition of a causal law to a set of fluents and define applicability as validity of the conjunction of these fluents in the state at hand. It is however natural and straightforward to consider arbitrary fluent formulae (c.f. Definition 7.2) instead.

describe the result of applying a set of causal laws $\mathcal{L}$ to some state s:

$$Trans(\mathcal{L}, s) \stackrel{\text{def}}{=} \Big(\big(s \setminus \bigcup_{\ell \in \mathcal{L}} inert^-(\ell) \big) \setminus \mathcal{F}_t \Big) \cup \bigcup_{\ell \in \mathcal{L}} inert^+(\ell) \cup \bigcup_{\ell \in \mathcal{L}} transient^+(\ell)$$

where $\mathcal{F}_t$ denotes the set of transient fluents considered in the dynamic system at hand. Hence, first all inert fluents are removed from s that are supposed to become false by some causal law in $\mathcal{L}$; afterwards, all transient fluents are removed; and finally, all inert and all transient fluents are added that are supposed to become true by some law in $\mathcal{L}$.

Example 3 Consider the two causal laws

$$\{shoot, loaded\} : \langle \{loaded\}, \{\}, \{bang, bullet\} \rangle \quad (3)$$

$$\{bullet, alive\} : \langle \{alive\}, \{\}, \{\} \rangle \quad (4)$$

whose conditions are contained, hence satisfied, in the state $s = \{alive, loaded, bullet, shoot\}$ (where *bullet* might result from a previous shot). We obtain $Trans(\{(3), (4)\}, s) = \{bang, bullet\}$. ∎

It is of course important to take into account the possibility that the simultaneous occurrence of two or more actions (or events) might have different effects than their separate occurrence. As an example, consider a table with a glass of water on it. Lifting the table on either side causes the water to be spilled whereas nothing similar happens if it is lifted simultaneously on opposite sides [9]. This situation may be specified by these three causal laws:

$$\{lift\text{-}left\} : \langle \{\}, \{\}, \{water\text{-}spills\} \rangle \quad (5)$$

$$\{lift\text{-}right\} : \langle \{\}, \{\}, \{water\text{-}spills\} \rangle \quad (6)$$

$$\{lift\text{-}left, lift\text{-}right\} : \langle \{\}, \{\}, \{\} \rangle \quad (7)$$

where *lift-left* and *lift-right* both are action fluents and *water-spills*, too, is transient (but not an action). Unfortunately each of these laws is applicable in the state $s = \{lift\text{-}left, lift\text{-}right\}$, thus determining the unintended result $Trans(\{(5), (6), (7)\}, s) = \{water\text{-}spills\}$.

In order to avoid such conclusions, we employ an additional criterion to suppress the application of some causal law as soon as, informally speaking, more specific information is available (see also [3, 13]). For instance, causal law (7) should override both (5) and (6) whenever it applies. A given set of causal laws naturally determines the following partial relation on the elements:

Definition 7.4 Let ℓ_1, ℓ_2 be two causal laws then ℓ_1 is called *more specific* than ℓ_2, written $\ell_1 \prec \ell_2$, iff $cond(\ell_1) \supset cond(\ell_2)$.

E.g., (7) $\prec$ (5) and (7) $\prec$ (6) but neither (5) $\prec$ (6) nor (6) $\prec$ (5).[6]

Based on the specificity criterion, the causal model of a dynamic system is obtained from a set of causal laws as follows:

[6] If we allow for arbitrary fluent formulae as conditions of causal laws (c.f. Footnote 5), then a law with condition c_1 is said to be more specific than a law with condition c_2 iff $\forall s.\, s \models c_1 \rightarrow c_2$ and $\exists s.\, s \models \neg(c_2 \rightarrow c_1)$.

Definition 7.5 Let $\mathcal{F}$ be a set of fluents and $\mathcal{L}$ a set of causal laws. For each consistent state $s \subseteq \mathcal{F}$ we define

$$\mathcal{L}(s) := \{\, \ell \in \mathcal{L} \mid \mathit{cond}(\ell) \subseteq s \;\&\; \neg\exists \ell' \in \mathcal{L}\,.\; \ell' \prec \ell \;\&\; \mathit{cond}(\ell') \subseteq s \,\}\,.$$

Then $\Phi(s) := \mathit{Trans}(\mathcal{L}(s), s)$.

Put in words, $\mathcal{L}(s)$ contains all causal laws $\ell \in \mathcal{L}$ that are applicable in s (i.e., which satisfy $\mathit{cond}(\ell) \subseteq s$) unless there is a more specific law $\ell' \in \mathcal{L}$ that is also applicable (i.e., that satisfies both $\ell' \prec \ell$ and $\mathit{cond}(\ell') \subseteq s$). For instance, since the two causal laws (5) and (6) are less specific than (7), we now obtain—due to $\mathcal{L}(\{\mathit{lift\text{-}left}, \mathit{lift\text{-}right}\}) = \{\ell_3\}$—the successor state $\Phi(\{\mathit{lift\text{-}left}, \mathit{lift\text{-}right}\}) = \{\}$, as intended. On the other hand, we still obtain, say, $\Phi(\{\mathit{lift\text{-}left}\}) = \{\mathit{water\text{-}spills}\}$ because law (7), though more specific than (5), does not apply here.

One should be aware of the fact that nonetheless it might well happen that two most specific applicable laws have mutually exclusive effects. A reasonable way to handle this problem will be proposed and formalized below, in Section 9. For the moment we assume that the combination of most specific causal laws never gives rise to a conflict, that is,

$$\bigcup_{\ell \in \mathcal{L}(s)} \mathit{inert}^-(\ell) \;\cap\; \bigcup_{\ell \in \mathcal{L}(s)} \mathit{inert}^+(\ell) \;=\; \{\} \tag{8}$$

for each (consistent) state s (where $\mathcal{L}(s)$ is as in Definition 7.5).

7.3 A Model Structure

The specification of state transition can be used to describe the evolution of a dynamic system over a longer period and under the influence of one or more agents. To direct the development of a system, these agents are able to (simultaneously) execute actions. The performance of one or more actions in a particular state is modeled by adding the corresponding set of action fluents to the state descriptions prior to applying the transition function. Since we take action fluents as transient, these are immediately removed automatically during a state transition.

Definition 7.6 Let $(\mathcal{F}, \Phi)$ be a dynamic system with action fluents $\mathcal{F}_a \subset \mathcal{F}$, and let $\sigma = [a_1, \ldots, a_n]$ $(n \geq 0)$ be a sequence of sets of action fluents (i.e., $a_i \subseteq \mathcal{F}_a$). Suppose s_0 is a consistent state, then the *application* of σ to s_0 yields an infinite sequence of system states $\langle s_1, \ldots, s_n, s_{n+1}, \ldots \rangle$ where

- $s_1 = s_0 \cup a_1$;
- $s_{i+1} = \Phi(s_i) \cup a_{i+1}$, for each $1 \leq i < n$; and
- $s_{i+1} = \Phi(s_i)$ for each $i \geq n$

provided each of $s_1, \ldots, s_n$ is consistent—otherwise the application of σ to s_0 is undefined. If it is defined then the triple $(\sigma, s_0, \langle s_1, \ldots \rangle)$ is a *development* in $(\mathcal{F}, \Phi)$.

Notice that some sets of actions a_i might be empty, that is, the agents have the possibility to pause for a moment and let the system evolve autonomously. Notice further that, after having executed the entire sequence of actions, the resulting state is not necessarily stable; the system might run into a limit cycle by oscillating among a number of states. Although a transition function Φ should be designed such that no inconsistent state results from a consistent one (c.f. Definition 7.1), the process of adding action fluents may cause inconsistency of some state s_i. This is the reason for the additional requirement of consistency.

Example 4 Recall our formalization of the Yale Shooting domain with the causal model being determined by the two causal laws (3) and (4). The application of the sequence $[\{\}, \{shoot\}]$ to the initial state $s_0 = \{bang, alive, loaded\}$ yields

$$\begin{array}{rcl} s_1 & = & s_0 \cup \{\} \;=\; \{bang, alive, loaded\} \\ s_2 & = & \Phi(s_1) \cup \{shoot\} \;=\; \{alive, loaded, shoot\} \\ s_3 & = & \Phi(s_2) \;=\; \{alive, bang, bullet\} \\ s_4 & = & \Phi(s_3) \;=\; \{\} \\ s_5 & = & \Phi(s_4) \;=\; \{\} \\ & \vdots & \end{array}$$

■

A dynamic system that evolves over time may admit observations regarding the truth value of certain state components at particular timepoints. Such an observation can be formalized as a fluent formula associated with a particular timepoint. We then call a development (in the sense of Definition 7.6) a *model* of an observation iff the corresponding fluent formula is true at that time, that is, in the corresponding state:

Definition 7.7 Let $(\mathcal{F}, \Phi)$ be a dynamic system. An expression $[i]\psi$ is called an *observation* if $i \in \mathbb{N}$ and ψ is a fluent formula. This observation *holds* in a development $(\sigma, s_0, \langle s_1, \ldots \rangle)$ iff $s_i \models \psi$. A *model* of a set Ψ of observations is a development in $(\mathcal{F}, \Phi)$ where each element of Ψ holds.

For instance, $[1]alive \wedge \neg bullet$ and $[3]\neg alive$ are two observations that can be formulated in our Yale Shooting domain. This development is a model of these observations:

$$\begin{array}{l} (\,[\{shoot\}]\,,\, \{alive, loaded\}\,, \\ \quad \langle \{alive, loaded, shoot\}, \{alive, bang, bullet\}, \{\}, \{\}, \ldots \rangle\,) \end{array}$$

For we have $s_0 = \{alive, loaded\} \models alive \wedge \neg bullet$ and $s_3 = \{\} \models \neg alive$. The reader is invited to verify that not only in this development but in every model of our two observations the additional observation $[1]shoot \wedge loaded$ holds—hence, we are allowed to conclude that a *shoot* action must have taken place and that the gun was necessarily loaded at the beginning.

An observation that holds in all models of a set of observations is considered to be entailed by the latter:

Definition 7.8 Let $(\mathcal{F}, \Phi)$ be a dynamic system and Ψ a set of observations. Ψ *entails* an additional observation $[i]\psi$, written $\Psi \models_\Phi [i]\psi$, iff $[i]\psi$ holds in each model of Ψ.

Based on this model-theoretic formalization of dynamic systems, we can classify the different modes of reasoning mentioned at the beginning.

- A *temporal projection* problem consists of an initial state s_0 along with a sequence of sets of actions $\sigma = [a_1, \ldots, a_n]$. The task is to compute the resulting state after having applied σ to s_0.

  In terms of our theory, the problem is essentially to find a model for the particular set of observations that describes the given initial state and exactly those occurrences of action fluents which are determined by σ, i.e.,

$$\begin{aligned}\Psi \;=\; \{\, & [0] \textstyle\bigwedge_{f \in s_0} f \wedge \bigwedge_{f \in \mathcal{F} \setminus s_0} \neg f\,, \\ & [1] \textstyle\bigwedge_{f \in a_1} f \wedge \bigwedge_{f \in \mathcal{F}_a \setminus a_1} \neg f\,, \\ & \vdots \\ & [n] \textstyle\bigwedge_{f \in a_n} f \wedge \bigwedge_{f \in \mathcal{F}_a \setminus a_n} \neg f \,\}\end{aligned}$$

  where $\mathcal{F}_a$ denotes the underlying action fluents.

- A *classical planning* problem consists of an initial state s_0 and a fluent formula g, called the *goal*. The task is to find a sequence of sets of actions σ, a *plan*, whose application to s_0 yields a sequence of system states containing one particular state s_n which satisfies g.

  In terms of our theory, the problem is essentially to find a model for this set of observations:

$$\begin{aligned}\Psi \;=\; \{\, & [0] \textstyle\bigwedge_{f \in s_0} f \wedge \bigwedge_{f \in \mathcal{F} \setminus s_0} \neg f\,, \\ & [n] g \,\}\end{aligned}$$

  for some $n \in \mathbb{N}$.

- A *postdiction* problem consists of a narrative given by a set of observations Ψ along with a sequence of sets of actions $\sigma = [a_1, \ldots, a_n]$. The task is to decide whether an additional observation is a logical consequence of this scenario.

  In terms of our theory, the problem is essentially to decide entailment wrt the particular set of observations that includes the given ones and describes exactly those occurrences of action fluents which are determined by σ, i.e.,

$$\begin{aligned}\Psi \;\cup\; \{\, & [1] \textstyle\bigwedge_{f \in a_1} f \wedge \bigwedge_{f \in \mathcal{F}_a \setminus a_1} \neg f\,, \\ & \vdots \\ & [n] \textstyle\bigwedge_{f \in a_n} f \wedge \bigwedge_{f \in \mathcal{F}_a \setminus a_n} \neg f \,\}\,.\end{aligned}$$

8 A Declarative Semantics for TEAL

We now take the Logic of Proactive Environments (LPE, for short) of the preceding section as the basis for a declarative semantics for our action language, TEAL. This semantics goes beyond the mere operational semantics in supporting, e.g., reasoning backwards to explain how a particular configuration came about.

Consider a given TEAL specification. This we interpret by a dynamic system as follows. The set of action fluents $\mathcal{F}_a$ consists of all (transient) fluents $exec(a)$ where a is an action term. All other fluents, inert or transient, are adopted. The major challenge when employing LPE as a semantics for TEAL specifications is to find a suitable interpretation of action rules in terms of causal laws. Notice that causal laws, as opposed to action rules, have no temporal structure; they only concern the state at the time of their application, and the one that immediately follows. Reference to conditions in the past and effects expected in the farther future, respectively, can however be made by introducing additional transient fluents, separately for each action rule. These fluents might be regarded as a flags indicating that the action rule in question is 'progressed.'

Consider an action rule r,

$$\bigwedge_i \bullet^i c_i \wedge exec(a) \rightarrow \bigwedge_j \bigcirc^j e_j \tag{9}$$

Let $n_\alpha, n_\omega \in \mathbb{N}$ so that n_α is the maximal number of nested $\bullet$ and n_ω the maximal number of nested $\bigcirc$s in this rule. Then r is re-written as

$$\bigwedge_{c\in C_{n_\alpha}} \bullet^{n_\alpha} c \wedge \ldots \wedge \bigwedge_{c\in C_0} \bullet^0 c \wedge exec(a) \;\rightarrow\; \bigwedge_{e\in E_1} \bigcirc^1 e \wedge \ldots \wedge \bigwedge_{e\in E_{n_\omega}} \bigcirc^{n_\omega} e$$

where each C_i is the (possibly empty) set consisting of all the condition literals following i $\bullet$, and where each E_j is the (possibly empty) set consisting of all the effect literals following i $\bigcirc$s. Now we introduce $n_\alpha+n_\omega-1$ new transient fluents $f^r_{-n_\alpha}, \ldots, f^r_{-1}, f^r_1, \ldots, f^r_{n_\omega-1}$. With their help, action rule (9) is interpreted by the following $n_\alpha + n_\omega + 1$ causal laws:

$$\begin{array}{rcl}
\bigwedge_{c\in C_{n_\alpha}} c & : & \langle\{\},\{\},\{f^r_{-n_\alpha}\}\rangle \\
\bigwedge_{c\in C_{n_\alpha-1}} c \wedge f^r_{-n_\alpha} & : & \langle\{\},\{\},\{f^r_{-n_\alpha+1}\}\rangle \\
& \vdots & \\
\bigwedge_{c\in C_1} c \wedge f^r_{-2} & : & \langle\{\},\{\},\{f^r_{-1}\}\rangle \\
\bigwedge_{c\in C_0} c \wedge f^r_{-1} \wedge exec(a) & : & \langle E^-_1, E^+_1, Et^+_1 \cup \{f^r_1\}\rangle \\
f^r_1 & : & \langle E^-_2, E^+_2, Et^+_2 \cup \{f^r_2\}\rangle \\
& \vdots & \\
f^r_{n_\omega-2} & : & \langle E^-_{n_\omega-1}, E^+_{n_\omega-1}, Et^+_{n_\omega-1} \cup \{f^r_{n_\omega-1}\}\rangle \\
f^r_{n_\omega-1} & : & \langle E^-_{n_\omega}, E^+_{n_\omega}, Et^+_{n_\omega}\rangle
\end{array} \tag{10}$$

where E^+_j and E^-_j denote all affirmative and negated, respectively, inert fluents in E_j, and Et^+_j contains all transient fluents which occur affirmatively in E_j. The extra transient fluents $f^r_{-n_\alpha}, \ldots, f^r_{n_\omega}$ serve as flags indicating that the action rule in question is

being 'processed', and each particular one of these flags denotes the current status of this execution.

The set of causal laws designed for an action rule reflect the latter as follows. Let $(\mathcal{F}, \models)$ be an execution model and $(\sigma, s_0, \langle s_1, s_2, \ldots\rangle)$ a development. At any timepoint t the antecedent of an action rule is true, i.e., $t \models \bigwedge_i \bullet^{n_i} e_i \wedge exec(a)$, iff beginning at time $t - n_\alpha$ the conditions of the first $n_\alpha + 1$ causal laws hold successively in the states $s_{t-n_\alpha}, \ldots, s_t$. Likewise, the consequent of a rule is true at time t, i.e., $t \models \bigwedge_j \bigcirc^j e_j$, iff the conditions of the $n_\omega - 1$ remaining causal laws hold successively in the states $s_{t+1}, \ldots, s_{t+n_\omega-1}$.

Definition 8.1 Consider a TEAL specification which consists of a set of fluents $\mathcal{F}$ and a set of action rules $\mathcal{R}$. Then $(\mathcal{F}, \mathcal{R})$ is *interpreted* by the dynamic system $(\widehat{\mathcal{F}}, \widehat{\Phi})$ where

- $\widehat{\mathcal{F}}$ is $\mathcal{F}$ augmented by the transient fluent needed to rewrite $\mathcal{R}$ as causal laws; and
- $\widehat{\Phi}$ is the causal model for the causal laws thus obtained.

Based on the notions of model and entailment of Section 7.3, our interpretation of TEAL action rules as causal laws provides a declarative semantics of specifications in TEAL. This semantics provably captures the operational semantics as described in Section 5 if no conflicts arise during the execution. Prior to establishing the formal proof, let us illustrate the application of LPE to TEAL by taking up again our Timesharing domain of Section 4.

The translation of the TEAL rules into LPE is as follows: Let the sets $\mathcal{F}_t$ and $\mathcal{F}_a$ of transient and action fluents, respectively, coincide and be as follows.

$$\begin{aligned}\mathcal{F}_t = \mathcal{F}_a = \; & \{exec(u\ takes\ r) : u \in \{u_1, u_2, os\},\ r \in \{r_1, r_2\}\} \\ & \cup \{exec(u\ drops\ r) : u \in \{u_1, u_2, os\},\ r \in \{r_1, r_2\}\} \\ & \cup \{exec(u\ succeeds) : u \in \{u_1, u_2\}\}\end{aligned}$$

and $\mathcal{F} = \mathcal{F}_a \cup \{u\ has\ r : u \in \{u_1, u_2, os\},\ r \in \{r_1, r_2\}\}$. Then $(T1) - (T4)$ give rise to a causal law each, for example for $(T1)$ we get:

$$(T1') \quad \neg u_1\ has\ r_1 \wedge \neg u_2\ has\ r_1 \wedge \neg os\ has\ r_1 \wedge exec(u_1\ takes\ r_1) : \langle\{\}, \{u_1\ has\ r_1\}, \{\}\rangle$$

Clauses $(T5) - (T6)$ similarly give rise to a causal law each, e.g. for $(T5)$ we get:

$$(T5') \quad u_1\ has\ r_1 \wedge u_1\ has\ r_2 \wedge exec(u_1\ succeeds) : \langle\{u_1\ has\ r_1, u_1\ has\ r_2\}, \{\}, \{\}\rangle$$

Finally, clause (DR) of Section 6 gives rise to the following causal law:

$$(DR') \quad u\ has\ r \wedge exec(u\ drops\ r) : \langle\{u\ has\ r\}, \{\}, \{\}\rangle$$

for any $u \in \{u_1, u_2, os\}$ and $r \in \{r_1, r_2\}$.

Recall the execution scenario $\sigma = [a_1, \ldots, a_{14}]$ of Section 6:

$$\begin{array}{l}[\ \{exec(u_1\ takes\ r_1)\}, \{exec(u_1\ takes\ r_2)\}, \{exec(u_1\ succeeds)\},\\ \ \{exec(u_2\ takes\ r_1), exec(os\ takes\ r2)\}, \{\}, \{\}, \{\}, \{\}, \{\}, \{\}, \{\},\\ \ \{exec(os\ drops\ r2)\}, \{exec(u_2\ takes\ r_2)\}, \{exec(u_2\ succeeds)\}\ ]\end{array}$$

Suppose the initial state of the database be such that all resources are available and no action occurs, i.e.,

$$s_0 \ = \ \{\neg\, u\ has\ r : u \in \{u_1, u_2, os\},\ r \in \{r_1, r_2\}\} \cup \{\neg f : f \in \mathcal{F}_a\}$$

State s_0 and action sequence $\sigma = [a_1, \ldots, a_{14}]$ admit a unique development $(\sigma, s_0, \langle s_1, s_2, \ldots\rangle)$, which, for instance, entails the following observations:

$$\begin{array}{rl}[3] & u_1\ has\ r_1 \ \wedge\ u_1\ has\ r_2\\ [7] & u_2\ has\ r_1 \ \wedge\ \neg\, u_2\ has\ r_2 \ \wedge\ os\ has\ r_2\\ [14] & u_2\ has\ r_1 \ \wedge\ u_2\ has\ r_2 \ \wedge\ exec(u_2\ succeeds)\end{array}$$

These observations illustrate that our declarative semantics captures the operational semantics of TEAL. In the following we will establish a formal proof of our declarative semantics being correct in this respect in general.

The proof requires some preparation. We call a *partial development* any initial segment $(\sigma, s_0, \langle s_1, \ldots, s_n\rangle)$ (where $n \geq 0$) of a development $(\sigma, s_0, \langle s_1, \ldots, s_n, s_{n+1}, \ldots\rangle)$.

Definition 8.2 An execution model $(\mathcal{F}, \models)$ is said to *coincide* with a partial development $(\sigma, s_0, \langle s_1, \ldots, s_n\rangle)$ iff

1. for all $t = 1, \ldots, n$ and all $f \in \mathcal{F}$ we have $t \models f$ iff $f \in s_t$, and
2. for all $f \in \widehat{\mathcal{F}} \setminus \mathcal{F}$ we have $f \notin s_0$.

The following observation shows that the causal laws by which we interpret an action rule, behave exactly the way the rule does.

Lemma 8.3 *Let $\mathcal{F}$ be a set of fluents, $\mathcal{R}$ a set of action rules, and $(\widehat{\mathcal{F}}, \widehat{\Phi})$ the dynamic system interpreting $(\mathcal{F}, \mathcal{R})$. Consider an execution model $(\mathcal{F}, \models)$ and a partial development $\delta = (\sigma, s_0, \langle s_1, \ldots, s_n\rangle)$ with which $(\mathcal{F}, \models)$ coincides. Then for any action rule $r \in \mathcal{R}$ of the form (9) and any $t \in \{1, \ldots, n-1\}$ such that $1 \leq t - n_\alpha < t + n_\omega \leq n$, we have*

$$t \ \models\ \bigwedge_i \bullet^i e_i \wedge exec(a) \wedge \bigwedge_j \bigcirc^j e_j$$

iff the corresponding causal laws of equation (10) are consecutively applicable in the states $s_{t-n_\alpha}, \ldots, s_t, \ldots, s_{t+n_\omega-1}$.

> **Proof:** Since $(\mathcal{F}, \models)$ coincides with δ, we know that $t \models \bigwedge_{c \in C_{n_\alpha}} c$ holds iff $C_{n_\alpha} \subseteq s_{t-n_\alpha}$. According to the topmost causal law in (10), the latter is equivalent to $f^r_{-n_\alpha} \in s_{t-n_\alpha+1}$.[7] Consequently, $t \models \bigwedge_{c \in C_{n_\alpha}} c \wedge \bigwedge_{c \in C_{n_\alpha - 1}} c$ iff both $C_{n_\alpha} \subseteq s_{t-n_\alpha}$ and $C_{n_\alpha - 1} \cup \{f^r_{-n_\alpha}\} \subseteq s_{t-n_\alpha+1}$. Reiterating this argument proves the claim.

[7] Notice that the newly introduced transient fluents are unique, that is, $f^r_{-n_\alpha}$, for instance, does not occur in any causal law but the two topmost ones in (10).

■

We are now prepared to prove that the declarative semantics obtained via LPE captures the operational semantics of TEAL, provided we do not encounter a conflict during execution.

Theorem 8.4 *Let $\mathcal{F}$ be a set of fluents, $\mathcal{R}$ a set of action rules, and $(\widehat{\mathcal{F}}, \widehat{\Phi})$ the dynamic system interpreting $(\mathcal{F}, \mathcal{R})$. Consider an execution scenario $\sigma = [a_1, \ldots, a_n]$ $(n \geq 0)$ and an initial state F_0 over $\mathcal{F}$ such that there is a unique execution model $(\mathcal{F}, \models)$ wrt. σ and with initial state F_0 updated by a_1. Then there is a unique development $\delta = (\sigma, s_0, \langle s_1, s_2, \ldots \rangle)$ in $(\widehat{\mathcal{F}}, \widehat{\Phi})$ satisfying $s_0 = F_0 \cup \{\neg f : f \in \widehat{\mathcal{F}} \setminus \mathcal{F}\}$. Moreover, $(\mathcal{F}, \models)$ coincides with all initial segment of δ.*

Proof: We prove by induction that there is a unique sequence of partial developments $(\sigma, s_0, \langle s_1 \rangle)$, $(\sigma, s_0, \langle s_1, s_2 \rangle)$, $(\sigma, s_0, \langle s_1, s_2, s_3 \rangle)$, ... such that $(\mathcal{F}, \models)$ coincides with each. Moreover, the limit of these initial segments is the unique development in $(\widehat{\mathcal{F}}, \widehat{\Phi})$ with initial state s_0 and execution scenario σ.

For the base case, let $s_1 = s_0 \cup a_1$. The initial state of the database in the execution model $(\mathcal{F}, \models)$ is given by F_0 updated by a_1. By construction of s_0 we thus have, for any $f \in \mathcal{F}$, that $1 \models f$ iff $f \in s_1$. Hence $(\mathcal{F}, \models)$ coincides with $(\sigma, s_0, \langle s_1 \rangle)$.

For the induction step, assume that $(\mathcal{F}, \models)$ coincides with the partial development $(\sigma, s_0, \langle s_1, \ldots, s_n \rangle)$ $(n \geq 1)$. Let $s_{n+1} = \widehat{\Phi}(s_n) \cup a_{n+1}$. We have to show, for any $f \in \mathcal{F}$, that $t + 1 \models f$ iff $f \in s_{n+1}$. To this end, consider an arbitrary fluent $f \in \mathcal{F}$. We distinguish three cases:

1. If $f \in a_{n+1}$, then both $t + 1 \models f$ and $f \in s_{n+1}$ by definition.
2. Otherwise, suppose there exists an action rule $r \in \mathcal{R}$ of the form (9) and some $t \leq n$ such that

$$t \models \bigwedge_i \bullet^i e_i \wedge exec(a) \wedge \bigwedge_j \bigcirc^j e_j$$

   and f (resp. $\neg f$) occurs in e_j with $t + j = n + 1$. Then $n + 1 \models f$ (resp. $n + 1 \models \neg f$). According to Lemma 8.3, this implies $f \in \Phi(s_n)$ (resp. $f \notin \Phi(s_n)$).
3. If there exists no such action rule, then f persists, that is, $n+1 \models f$ iff $n \models f$. According to Lemma 8.3, there is no causal law which is applicable in s_n and which has f as positive or negative effect. Consequently, $f \in \Phi(s_n)$ iff $f \in \Phi(s_n)$.

This proves that if $(\mathcal{F}, \models)$ coincides with $(\sigma, s_0, \langle s_1, \ldots, s_n \rangle)$, then so does it with $(\sigma, s_0, \langle s_1, \ldots, s_n, s_{n+1} \rangle)$.

The way in which we have constructed the successive states to obtain the sequence of partial developments is in accordance with Definition 7.6. Hence the limit δ of the sequence is the unique development in $(\widehat{\mathcal{F}}, \widehat{\Phi})$ wrt. σ and s_0.

■

9 Nondeterminism and Conflict Resolution in LPE

In this section, we extend our declarative semantics to so-called *nondeterministic* dynamic systems, which will then provide a basis also for handling conflicts. Nondeterminism arises if parts of the successor state are uncertain even in case the current state is completely known. This is reflected in the following definition, where the causal model consists of a relation on pairs of states (see, e.g., [4]) instead of a function as in Definition 7.1:

Definition 9.1 A *nondeterministic, propositional dynamic system* is a pair $(\mathcal{F}, \Phi)$ consisting of a set of fluents $\mathcal{F}$ and a relation $\Phi \subseteq 2^{\mathcal{F}} \times 2^{\mathcal{F}}$.

Given a state $s \subseteq \mathcal{F}$, each s' such that $(s, s') \in \Phi$ is called a *possible successor* state. A state is now said to be inconsistent in case it has no successor at all.

The concept of nondeterminism is reflected in an extended notion of a causal law where several effects $\langle i_i^-, i_i^+, t_i^+ \rangle$ can be associated with a single condition; each triple then determines a possible alternative:

Definition 9.2 Let $\mathcal{F} = \mathcal{F}_i \dot{\cup} \mathcal{F}_t$ be a set of inert and transient fluents. An *extended causal law* is a structure $c : \{\langle i_1^-, i_1^+, t_1^+ \rangle, \ldots, \langle i_n^-, i_n^+, t_n^+ \rangle\}$ where $n \geq 1$; $c \subseteq \mathcal{F}$; $i_i^-, i_i^+ \subseteq \mathcal{F}_i$; and $t_i^+ \subseteq \mathcal{F}_t$ $(1 \leq i \leq n)$.

Example 5 The Russian Turkey scenario (see, e.g., [21]) is obtained from the Yale Shooting domain by adding an action fluent *spin*. The effect of spinning its cylinder is that the firearm becomes randomly loaded or not, regardless of its state before. This nondeterministic effect is formalized by the following extended causal law:

$$\{spin\} : \left\{ \begin{array}{l} \langle \{loaded\}, \{\}, \{\} \rangle, \\ \langle \{\}, \{loaded\}, \{\} \rangle \end{array} \right\} \tag{11}$$

■

As for the special case of deterministic systems, the combination of all most specific laws shall determine the behavior of the system at hand. Hence, for each state s we define

$$\mathcal{L}(s) := \{\, \ell \in \mathcal{L} \mid \; cond(\ell) \subseteq s \;\; \& \;\; \neg \exists \ell' \in \mathcal{L}\,.\; \ell' \prec \ell \;\&\; cond(\ell') \subseteq s \,\},$$

similar to Definition 7.5, where $\mathcal{L}$ denotes the underlying set of (extended) causal laws. Now, suppose that $\mathcal{L}(s)$ be $\{c_1 : \mathcal{A}_1, \ldots, c_k : \mathcal{A}_k\}$ $(k \geq 0)$ and define

$$Poss(\mathcal{L}(s)) := \Big\{ \{c_1 : a_1, \ldots, c_k : a_k\} \;\Big|\; a_i \in \mathcal{A}_i \,(1 \leq i \leq k) \Big\}$$

That is, $Poss(\mathcal{L}(s))$ contains all combinations of alternatives. Each element then determines a possible successor state of s, i.e.,

$$(s, s') \in \Phi \quad \text{iff} \quad \exists \mathcal{P} \in Poss(\mathcal{L}(s)).\; s' = Trans(\mathcal{P}, s)\,.$$

Suppose, for example, (11) is the only applicable causal law in the state $s = \{alive, spin\}$. Then $Poss(\mathcal{L}(s))$ is

$$\Big\{\{spin : \langle\{loaded\}, \{\}, \{\}\rangle\}\,,\ \{spin : \langle\{\}, \{loaded\}, \{\}\rangle\}\Big\}$$

Hence both $(s, \{alive\}) \in \Phi$ and $(s, \{alive, loaded\}) \in \Phi$.

The concept of nondeterminism provides us with an interesting solution to the problem of concurrently performed actions with mutually exclusive effects. Consider, for instance, the two causal laws

$$\begin{array}{l} \{push\text{-}door\} : \langle\{\}, \{open\}, \{\}\rangle \\ \{pull\text{-}door\} \ : \langle\{open\}, \{\}, \{\}\rangle \end{array} \tag{12}$$

where *push-door* and *pull-door* denote action fluents while the inert fluent *open* describes the state of the door under consideration here. Now, assume three agents acting concurrently: The first one tries to push the door, the second one tries to pull it, and the third agent intends to lift the left hand side of a table inside the room (c.f. (5)–(7)). Suppose further that the door is currently closed and no water spills out of the glass placed on top of the table, that is, $s = \{push\text{-}door, pull\text{-}door, lift\text{-}left\}$. Now, aside from (5) both causal laws in (12) are applicable. However, the first one claimes the door be open in the succeeding state ($open \in i^+$) while the second one postulates the opposite ($open \in i^-$). Hence, our consistency condition, (8), is not satisfied here.

Most classical AI formalizations of concurrent actions, such as [15, 3, 11], declare situations like s inconsistent and, hence, do not allow any conclusions whatsoever about the successor state. It is impossible indeed that both actions *push-door* and *pull-door* succeed. However, as we have argued in [4] (in the context of a theory developed in [8] and extended in [3]), it is reasonable to draw conclusions at least about uninvolved fluents; e.g., we would like to conclude that the third agent is successful in lifting the table, which causes water be spilled. Preventing global inconsistency in case of local conflicts is the basic intention of this idea.

The notion of nondeterminism provides us with a ready approach to support such conclusions. Instead of declaring undefined the successor state of s, we take only the disputed fluents (here: *open*) as uncertain while any other effect (here: *water-spills*, coming from (5)) occurs as intended. In our example, we thus obtain two possible successors of s, namely, $\{open, water\text{-}spills\}$ and $\{water\text{-}spills\}$—providing us with the conclusion that *water-spills* become true.

This strategy of conflict solving is integrated in the following definition of how to obtain the causal model in case of nondeterministic systems:

Definition 9.3 Let $\mathcal{F}$ be a set of fluents and $\mathcal{L}$ a set of (extended) causal laws. For each (consistent) state $s \subseteq \mathcal{F}$ we define

$$\mathcal{L}(s) \ := \ \{\, \ell \in \mathcal{L} \mid \ cond(\ell) \subseteq s \ \ \&\ \ \neg\exists \ell' \in \mathcal{L}\,.\ \ell' \prec \ell \ \&\ cond(\ell') \subseteq s \,\}\,.$$

Suppose $\mathcal{L}(s)$ be $\{c_1 : \mathcal{A}_1, \ldots, c_k : \mathcal{A}_k\}$ $(k \geq 0)$ and let

$$Poss(\mathcal{L}(s)) \ := \ \Big\{\{c_1 : a_1, \ldots, c_k : a_k\} \ \Big|\ a_i \in \mathcal{A}_i\,(1 \leq i \leq k)\Big\}$$

Furthermore, we define $(s, s') \in \Phi$ iff

$$\exists \mathcal{P} \in \mathit{Poss}(\mathcal{L}(s)),\ i^{\swarrow} \in \mathit{Confl}(\mathcal{P}).\ \ s' = \mathit{Trans}(\mathcal{P}, s) \setminus i^{\swarrow}$$

where

$$\mathit{Confl}(\mathcal{P}) \ := \ \bigcup_{\ell \in \mathcal{P}} \mathit{inert}^{-}(\ell) \ \cap \ \bigcup_{\ell \in \mathcal{P}} \mathit{inert}^{+}(\ell)\,.$$

The set $\mathit{Confl}(\mathcal{P})$ is intended to contain all disputed fluents (c.f. (8)), and each possible combination of these fluents determines a possible successor state.[8]

Finally, the semantics developed in the previous section is extended to nondeterministic dynamic systems in the following way (c.f. Definition 7.6):

Definition 9.4 Let $(\mathcal{F}, \Phi)$ be a nondeterministic dynamic system with action fluents $\mathcal{F}_a \subset \mathcal{F}$, and let $\sigma = [a_1, \ldots, a_n]$ $(n \geq 0)$ be a sequence of sets of action fluents in $\mathcal{F}_a$. Suppose s_0 is a consistent state, then a triple $(\sigma, s_0, \langle s_1, \ldots, s_n, s_{n+1}, \ldots \rangle)$ is a *development* iff

- $s_1 \ = \ s_0 \cup a_1$;
- $s_{i+1} = s_i' \cup a_{i+1}$, where $(s_i, s_i') \in \Phi$, for each $1 \leq i < n$;
- $(s_i, s_{i+1}) \in \Phi$ for each $i \geq n$

and each of $s_1, \ldots, s_n$ is consistent.

10 Related Work

A well-known framework for reasoning about action and time is the one put forward by J.F.Allen in his article [1], which was seminal. Allen's theory is based on the notion of time intervals, and through a *meets* relation on pairs of intervals, time is structured linearly.

A critical examination by A.Galton [7] found Allen's framework unsuitable for representing facts about continuous change, and proposed revisions to accommodate this. In particular, the temporal ontology is enriched by instants in addition to intervals, and the property concept is refined into states of position and states of motion, respectively. As a result, Allen's category of processes is subsumed by Galton's category of properties.

In [2], Allen and Ferguson compare interval temporal logic with other formalisms, notably those based on the situation calculus, and present an approach to proactive environments based on explanation closure. For each property it is specified which events can change it. External events are divided into three classes;

- triggered events, which do not occur unless specifically triggered by an agent's action
- definite events, which are certain to occur, but at uncertain times, and

[8] Notice that we are supposed to remove the set $i^{\swarrow}$ from $\mathit{Trans}(\mathcal{P}, s)$, for all elements in $\mathit{Confl}(\mathcal{P})$ are first of all added when computing $\mathit{Trans}(\mathcal{P}, s)$.

- spontaneous events, which may or may not occur.

Reasoning about change then amounts to reasoning about which events may or may not have occurred, and it is seen that all events have to be accounted for in advance, even external ones.

By comparison, our approach does not necessitate any a priori tabulation of the system properties affected by external events. Rather, unforeseen effects caused by the environment are tackled on the fly, through dynamic satisfaction of constraints.

TEAL falls broadly within the MetaTem paradigm of executable temporal logic. The most important difference between TEAL and the MetaTem framework has to do with the scope of influence of the environment, compared to that of the agent. MetaTem requires the set of fluents controlled by the environment to be disjoint from the set of fluents controlled by the agent, cfr section 3.2.1 of [19], whereas the opposite requirement is made by TEAL. Here these sets are taken to coincide, or at least have a non-empty intersection, and the complications arising from the interference by the environment are of particular interest.

GOLOG is a logic programming language for dynamic domains, based on situation calculus semantics. Actions play a fundamental role, and assumptions about the environment are formally specified within the language, cfr. the discussion in section 6 of [16].

The interpreter for the GOLOG language maintains an explicit model of the system's environment, treating it as a logical database to be queried and reasoned with at runtime. This is reminiscent of our view of the evolving state of execution, but TEAL differs from GOLOG when it comes to computing a new state based on existing information and a given program of clauses. Where GOLOG utilises a simple solution to the frame problem put forward by Reiter [20] to deal with effects of the environment, no such harnessing is attempted by TEAL. Rather, arbitrary interference by the environment is expected and dealt with on the fly as explained in [19] and in section 3 of this paper.

11 Conclusion and further work

We have presented a temporal-logical language to model agents situated in proactive environments, and we have developed an abstract, general theory of such dynamic systems. We have illustrated that our framework allows for a natural treatment of concurrent actions and simultaneous events as well as delayed effects, and we have integrated a reasonable way to handle the problem of concurrent actions with mutually exclusive effects. We have then devised a formal semantics for TEAL programs as sets of LPE causal laws.

Delayed effects constitutes a particular category of indirect effects of actions. The *Ramification Problem* [10] deals with indirect effects that are so closely connected to the their triggering effects that they arise in the same state transition step. Accounting for such effects, which derive from state constraints, is amenable to the techniques of [22], and this issue is receiving further attention in ongoing work by the authors.

References

[1] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[2] J. F. Allen and G. Ferguson. Actions and events in interval temporal logic. *J. Logic and Computation*, 4(5), 1994.

[3] Chitta Baral and Michael Gelfond. Reasoning about effects of concurrent actions. *Journal of Logic Programming*, 31(1–3):85–117, 1997.

[4] S.-E. Bornscheuer and M. Thielscher. Explicit and implicit indeterminism: Reasoning about uncertain and contradictory specifications of dynamic systems. *Journal of Logic Programming*, 31(1–3):119–155, 1997.

[5] D. Gabbay and R. Nossum. A temporal executable agent language. *Workshop on Executable Temporal Logic at IJCAI95, Montreal*, 1995.

[6] Dov Gabbay, Ian Hodkinson, and Mark Reynolds, editors. *Temporal Logic: Mathematical Foundations and Computational Aspects*, volume 1. Oxford University Press, 1994.

[7] A. Galton. A critical examination of allen's theory of action and time. *Artificial Intelligence*, 42:159–188, 1990.

[8] Gelfond and Lifschitz. Representing action and change by logic programs. *JLP*, 17:301–321, 1993.

[9] Michael Gelfond, Vladimir Lifschitz, and Arkady Rabinov. What are the limitations of the situation calculus? In S. Boyer, editor, *Automated Reasoning, Essays in Honor of Woody Bledsoe*, pages 167–181. Kluwer Academic, 1991.

[10] Matthew L. Ginsberg and David E. Smith. Reasoning about action I: A possible worlds approach. *Artificial Intelligence*, 35:165–195, 1988.

[11] Gerd Große. Propositional State-Event Logic. In C. MacNish, D. Peirce, and L. M. Peireira, editors, *Proceedings of the European Workshop on Logics in AI (JELIA)*, volume 838, pages 316–331. Springer, September 1994.

[12] Steve Hanks and Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33(3):379–412, 1987.

[13] Steffen Hölldobler and Michael Thielscher. Computing change and specificity with equational logic programs. *Annals of Mathematics and Artificial Intelligence*, 14(1):99–133, 1995.

[14] Vladimir Lifschitz and Arkady Rabinov. Things that change by themselves. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 864–867, Detroit, MI, 1989.

[15] Fangzhen Lin and Yoav Shoham. Concurrent actions in the situation calculus. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 590–595, San Jose, CA, 1992. MIT Press.

[16] Levesgue Reiter Lesperance Lin and Scherl. Golog: a logic programming language for dynamic domains. *J. Logic Programming*, 19, 1994.

[17] P.J. McBrien. Principles of implementing historical databases in RDBMS. In *Proceedings of the 11th British National Conference on Databases.* Springer LNCS, 1993.

[18] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[19] Barringer Fisher Gabbay Owens and Reynolds. *The Imperative Future. Volume 1: Principles. Draft.* Wiley and Sons Inc, 1995.

[20] Raymond Reiter. *The Frame Problem in the Situation Calculus, A Simple Solution (Sometimes) and a Completeness Result for Goal Regression*, pages 359–380. Academic Press, San Diego,CA, 1991.

[21] E. Sandewall. *Features and Fluents. Volume 1: The representation of knowledge about dynamical systems.* Oxford University Press, 1994.

[22] Michael Thielscher. Ramification and causality. *Artificial Intelligence*, 89(1–2):317–364, 1997.