Relating high-level and low-level action descriptions in a logic of actions and change

Erik Sandewall[1]

Department of Computer and Information Science
Linköping University, S-58183 Linköping, Sweden

Abstract. We address the problem of formally proving high-level effect descriptions of actions from low-level operational definitions. The low-level operational description has a hybrid character, involving both continuous and discrete change; the effect description is wholly or predominantly discrete. Both descriptions are expressed in a logic of actions and change, with extensions for characterizing continuous change, discontinuities, the distinction between true and estimated values of state variables, and the distinction between success and failure of an action. Both descriptions also require the use of nonmonotonicity in the logic in question. The transition from operational definition to effect description furthermore involves the creation of a closure with respect to the set of possible ways that the action can fail.
We outline, by means of an example, how these issues can be addressed as an extension of existing results on logics of actions and change.

1 Background and topic

The area of *cognitive robotics* is a branch of artificial intelligence (AI), and attempts to develop formal methods for the design of robots that reason about their own actions. The capability to reason about continuously changing properties of the world and about the properties and imperfections of sensors and actuators is important for this goal. However, other issues have so far received much more attention in this research.

It is generally recognized, in AI, that one has a need for high-level descriptions of actions which describe the effects of a robot action in terms of its preconditions and postconditions. These high-level descriptions are required for the purpose of planning sequences of actions which will achieve desired goals, and for the modification of those sequences (replanning). It is generally recognized, as well, that one must separate the main action description, which characterizes its normal conditions and effects, from the additional information describing various exceptional cases.

Finally, it is generally recognized that non-monotonic logics are needed in order to obtain that the action descriptions be concise and well structured.

We observe, however, that the *effect descriptions* of actions which are used e.g. for planning are rarely directly executable, if ever; they must be complemented by low-level *operational definitions* which specify how the action is actually performed in terms of continuous change, sensors, actuators, and control algorithms. In view of that observation, the present article addresses the following problem: *how can high-level effect descriptions of actions be formally proved from their low-level operational definitions.*

The present paper will describe our solution to this problem in terms of an example. Other, concurrent articles describe it in more general terms. The following concrete example will be used. Consider the move operation for a robot with the following properties: If the robot is told to move forward, it will gradually increase its velocity until it reaches 1 m/s, and then proceed at constant speed. The acceleration is a constant 0.5 m/s^2 until it reaches maximum speed. If the robot is told to stop, it stops instantly. There are also two cases where it will stop under the command of its motoric system, regardless of higher-level control: if its proximity sensor indicates an obstacle right in front of it, and if the emergency stop system which has been installed for safety reasons in the robot hall broadcasts a command to stop. The emergency stop is caused when any one of several alarm buttons is pushed; the stop request is then sent over an IR channel.

The relatively easy part of the problem is to arrange that the effect description for the normal case of the action follows from the operational definition. The hard part is to arrange that the auxiliary information which characterizes the failure of the action (for example because of power failure, obstacle in the way, etc) is represented in the context of the operational definition and the effect description, and that the latter kind of auxiliary information can be derived from the former one.

Although the action in the chosen example is an elementary one, we shall also reference related work showing how composite and higher-level actions can be defined in terms of elementary ones, in such a fashion that the same verification technique applies to them as well.

2 Formalism

Our approach will be to represent both the low-level and the high-level description of the action in terms of a first-order predicate calculus (FOPC) with several sorts, where time, represented as the real numbers, is one of those sorts. This is the traditional approach in AI and in this branch of research. Two primary predicates are used, H for *Holds* and D for *Do*, and they are defined as follows. $\mathsf{H}(t, p)$ says that the "propositional fluent" or "propositional feature" p holds at time t. In other words, p is reified and $\mathsf{H}(t, p)$ is the same as $p(t)$ in the case where p is atomic. (One reason for the reification is that some additional, auxiliary predicates also take such fluents or features as an argument). $\mathsf{D}([s, t], a)$ says that the action a is performed over the time interval $[s, t]$.

A "feature" is then what is called a "state variable" in engineering; the word "variable" is avoided here because it has another connotation in logic. Non-propositional features are also admitted, using the notation $\mathsf{H}(t, f\!:\!v)$ where $f\!:\!v$

is the proposition saying that the feature f has the value v.

We proceed now to describe which extensions to these common-sense notions are needed for the present purpose.

2.1 Separate Holds predicates for actual and estimated feature values

The operational definition for the robot shall not be expressed directly in terms of quantities in the world itself, but in terms of entities that are known to the robot, in particular through its sensors. Thus, rather than just having one feature (state variable, fluent) e.g. for the robot's position, one needs to distinguish between the actual position and the position as estimated by the robot itself. Statements about the relationship between those two quantities express our knowledge of the accuracy of the position sensor; the operational definition of the move action will use the estimated value; and the effect description of the same action will use the actual, physical position. The characterization of actuators is analogous to the one for sensors. These different kinds of information are then combined when effect descriptions are derived from operational action definitions.

Although in principle it would be possible to introduce two distinct state variables whenever needed, we prefer the more systematic approach proposed by Drakengren (in our research group) of complementing the traditional *Holds* predicate with a second predicate for the estimated value of a feature. Thus, we distinguish two predicates: a predicate H which specifies what is true in the world, and a predicate B which has the same argument structure as H and which expresses what value the agent (robot) *believes* is assigned to each feature. For example,

$$\mathsf{H}(25.05, pos(r) : (45.2, 63.8))$$

will express that at time 25.05, the current value of the feature $pos(r)$ which is intended as the current position of the robot r in Cartesian room coordinates is $(45.2, 63.8)$. Similarly,

$$\mathsf{B}(25.05, pos(r) : (45.4, 63.5))$$

will express that the estimated value for the same time is $(45.4, 63.5)$.

Since some feature values may be unknown, B generally allows nil as a feature value for the undefined value. For the moment, we do not allow for a set of alternative values in B, but this would be a natural generalization for the future.

Notice that B expresses the "raw" belief that the ego obtains from its sensors. It is possible that its reasoning capability will be used to question and revise some of those values, but that is not reflected in B as presently interpreted.

In this way we can describe how a sensor behaves relative to the world (B depends on H) and how an actuator behaves (H depends on B but also on context information).

2.2 Features for the current example

The following are the features (state variables) that are needed for the chosen simple application. For navigation, it is first of all necessary to specify the robot's attitude, that is, the direction that it is "looking".

$attid[r]$ shall be either of N, S, E, or W, indicating where r is "looking" and with the obvious interpretations. North is in the direction of the y axis, and east is in the direction of the x axis.

$pos[r]$ are the present coordinates of the robot r in the floorplane, represented as a complex number (x, y). Addition, subtraction, and multiplication of complex numbers have their standard definitions.

$hdiag[r]$ is the half diagonal of the robot r, viewed as a rectangular object, in its own coordinate system. That is, it is a pair (x, y) where $2x$ is the width of the object and $2y$ is its length.

$obstac[r]$ is true if the robot r is just runnig into an obstacle. In the context of B, this is detected by its obstacle detection sensor.

$emergency$ is true if an emergency button is being pushed.

The arguments of these feature functions will sometimes be enclosed by round parentheses, sometimes with square brackets, according to the following convention. $attid(r)$ is a *feature* which may appear in an argument of H ; $attid[r]$ is the *value* of $attid(r)$ in the current context. Thus,

$$\mathsf{H}(t, attid(r) : v) \leftrightarrow attid[r] = v$$

in a context where t is the current time.

Of these feature-valued functions, $attid[r]$ and $hdiag[x]$ for all mobile objects x are assumed to be always correctly known by the robot. This is expressed by

$$\mathsf{B}(t, attid(r) : v) \leftrightarrow \mathsf{H}(t, attid(r) : v)$$

and (illustrating at the same time how we would deal with the case where there are several kinds of rectangular objects, only some of which are mobile):

$$\forall x[\mathsf{Isa}(x, \mathsf{mobile}) \rightarrow (\mathsf{B}(t, hdiag(x) : v) \leftrightarrow \mathsf{H}(t, hdiag(x) : v))]$$

with implicit universal quantification over free variables, as usual. For pos we retain the possibility of a discrepancy between held and believed values.

2.3 Failible actions

For characterizing movement actions, it is necessary to have appropriate notation for dealing with the applicability, success, and failure of actions. We introduce several variants of the D ("Do") operator, namely one variant when the action succeeds and another one when it fails. The following are the details of the notation. In all cases, s and t are timepoints (usually s for starting time and t for termination time) and a is an action.

$\mathsf{G}(s, a)$: the action a is invoked ("go") at time s

$\mathsf{A}(s, a)$: the action a is applicable at time s

$\mathsf{D_s}([s,t],a)$: the action a is executed successfully over the time interval $[s,t]$: it starts at time s and terminates successfully at time t.

$\mathsf{D_f}([s,t],a)$: the action a is executed but fails over the time interval $[s,t]$: it starts at time s and terminates with failure at time t.

$\mathsf{D_c}([s,t],a)$: the action a is being executed; the execution started at time s and has not been terminated before time t. (It may terminate at t or later).

It is also convenient to have the following abbreviation:

$\mathsf{D_n}([s,t],a)$, defined as $\mathsf{D_c}([s,t],a) \wedge \neg \mathsf{D_s}([s,t],a)$.

For both $\mathsf{D_s}$ and $\mathsf{D_f}$, s is the time when the action was invoked, and t is the exact time when it completes with success or failure.

A set of axioms stating the properties and relationships of these predicates is included in the full version of the paper. (See the WWW reference at the end of this paper). The following is an example of such an axiom. If an action is being executed, then it must have been invoked:

$$\mathsf{D_c}([s,t],a) \rightarrow \mathsf{G}(s,a)$$

As another example of the notation, here is the formula stating that a condition φ guarantees that an action always succeeds:

$$\mathsf{H}(s,\varphi) \wedge \mathsf{G}(s,a) \rightarrow \exists t[\mathsf{D_s}([s,t],a)]$$

Ordinary action laws specify the action's effects when it succeeds. They are therefore written as usual and with $\mathsf{D_s}$ on the antecedent side: if preconditions apply and the action is performed successfully, then the postconditions result.

The full paper also contains an action schema for the description of actions in a particular special case, namely those actions whose effects are characterized by a precondition, a prevail condition, and a postcondition, and where the postcondition is at the same time the termination condition for the action. The prevail condition must be satisfied throughout the execution of the action; if it is violated then the action fails. The sections that follow below show how the formalism can be used for more general classes of actions.

3 Effect descriptions for actions

The effect descriptions shall be expressed in terms of what goes on in the world itself, and not in terms of internal quantities or beliefs in the robot. They are to consist of two parts: a *success description* which specifies the effects of the action if it succeeds, and a *failure description* which specifies the various ways that the action can fail.

The success description shall have the general form

$$\mathsf{D_s}([s,t],a) \rightarrow \text{(results)}$$

The applicability conditions need not be stated here, since they are implied by $\mathsf{D_s}$ according to the general axioms relating the above predicates.

One may consider two ways of writing the failure descriptions. A *closed* failure description has the general form

$$\mathsf{D_f}([s,t],a) \rightarrow \neg\mathsf{A}(s,a) \vee \delta_1 \vee \delta_2 \vee \ldots \vee \delta_k$$

where each of the δ_i characterizes one way or group of ways that the action may go wrong. An *open* failure description, on the other hand, is a set of axioms each of which has the form

$$\delta_i \rightarrow \mathsf{D_f}([s,t],a)$$

The difference between these alternatives is not simply a question whether additional causes of error are to be edited into the single axiom for the closed description, or added as an additional axiom in the open description. In particular, the open description makes it possible to have cases of failure which are *implied* by underlying or more general circumstances, so that it is not necessary to enumerate all error reasons explicitly.

For these reasons, open failure descriptions would seem to be much more in agreement with the tradition in A.I. We shall argue, however, that for many purposes it is necessary to have a closed failure description on the effect level (that is, the high-level description of the action), since only so will one obtain a firm knowledge of when the action is guaranteed to work correctly. For the operational definition, on the other hand, it is quite appropriate to use an open failure description. The derivation of a closed failure description for the high level from an open failure description on the lower level is therefore a part of the problem being addressed here.

One of the axioms that were referenced in section 2 states that if an action is invoked then it either succeeds or fails. Using that axiom, it follows that either the success description or the failure description must apply whenever an action has been invoked. Also, if one can prove that all of the δ_i in a closed failure description must be false and that the applicability condition holds, then one is entitled to conclude that the action succeeds if invoked.

In our particular example, we obtain the following effect descriptions. The success description is

$$\mathsf{D_s}([s,t],\textsc{goto}(r,p)) \rightarrow \mathsf{H}(t, pos(r)\!:\!p)$$

saying that if r goes to p successfully, then at the end of the action period it is at p. Under the assumptions outlined above, the failure description is

$$\mathsf{D_f}([s,t],\textsc{goto}(r,p)) \rightarrow \neg\mathsf{A}(s,a) \vee \mathsf{H}(t, obstac(r)) \vee \mathsf{H}(t, emergency)$$

The problem now is to show how these effect descriptions can be derived from an operational description.

4 Operational definition of the move action

We shall introduce the first set of formulae for the operational definition of a simple move action in the framework that has just been defined. We do it for the elementary action GOTO that just moves ahead to a given position or to a given object.

4.1 Auxiliary constructs

The basic features for describing the robot have already been introduced. For navigation, it is convenient to use the robot's coordinate system for many navigation tasks. The following auxiliary definitions do that.

$xc(z)$ and $yc(z)$ extract the x and y coordinates, respectively, from a complex number.

$mc(z, r)$ transforms the coordinate $z = (x, y)$ in floorplan coordinates to the coordinate system of the robot (or other mobile object) r, where y is straight ahead in the direction of the attitude of r, and x is towards the right. This function is defined as

$$mc(z, r) = (z - pos[r]) * cdir(attid[r])$$

where the auxiliary function $cdir$ is defined as

$$\{\mathsf{N} \mapsto (1, 0), \mathsf{S} \mapsto (-1, 0), \mathsf{E} \mapsto (0, 1), \mathsf{W} \mapsto (0, -1)\}$$

Notice that these are mathematical functions; they are not features and are not part of the temporal interpretations. These mathematical functions are of course assumed known to the robot, in the sense that they can be used in the definitions of the operational action definitions.

4.2 Applicability of the action

We proceed to using the functions that have so been introduced, and begin with the applicability condition of the action GOTO. It can be defined as follows.

$$\begin{gathered}\mathsf{A}(s, \text{GOTO}(r, p)) \leftrightarrow \\ \mathsf{B}(s,\ |xc(mc(p, r))| \leq xc(hdiag[r]) \wedge \\ yc(mc(p, r)) > 0 \wedge \\ \partial pos[r] = \partial^2 pos[r] = (0, 0))\end{gathered}$$

taking certain obvious liberties in the second argument of B. This definition means that the operation of going to the position p in an elementary way is applicable iff p is ahead of the robot r in its own forward direction, with a displacement towards the left or the right which is no more than the half-width of the robot. In other words, some point of the robot will touch p if the action is successful, but it will not necessarily be the centerline of the robot.

Notice that the applicability is defined in terms of what the robot believes, not in terms of what is actually the case. One may think of the action's operational definition as a software subroutine which is invoked in the way specified by the G predicate. This subroutine will first relate the parameters that are sent to it, to its present estimates of the state of the world, and if it considers that the requested action is impossible in the present situation, it will return immediately and without having engaged any actuator. This immediate return is modelled by the negation of A, and is assumed to take zero time. If the initial tests are passed, then the action is considered applicable in the sense of A, it takes non-zero time, and still it may either succeed or fail.

In this framework, it is evident that applicability must be *defined* in terms of "believed" rather than "held" values of features/fluents. Still, given that one has some knowledge about the relationship between believed and held values for each sensor (that is, about the properties of that sensor), one can of course expect to *derive* a condition for applicability in terms of "held" or actual values.

In the present, relatively simple approach, we assume that the robot has one specific belief about the position, and do not allow for the case that it only believes that the position is in a certain range.

4.3 Invocation and termination of the action

We now proceed to the invocation of GOTO. The general structure of the definitions is as follows: the proposition $\mathsf{G}(\text{GOTO}(r,p))$ shall imply a certain momentary, discrete change in the world, which starts a chain of spontaneous change corresponding to the execution of the action, and finally there is (if necessary) another momentary, discrete change when the action terminates. The important thing is that that what goes on while the action is in progress is modelled using a technique for describing physical movements.

The following intuition may be useful. Suppose the robot were not a cart that is propelled by motors driving its wheels, but instead just a particle that slides without friction on a flat surface. What is needed then is to assign a velocity to it at the beginning of the action, allow it to run its course according to the laws of physics, and then reassign its velocity again when the destination has been reached.

The concept of reassignment is intuitively clear, particularly from a programming language background, but it requires special treatment in a logic of the present kind. Let us first show how it is used, and then proceed to its formal properties in a later section. Writing $\mathsf{H}(t, f := v)$ for "the feature f adopts the value v at time t", the invocation of GOTO can be defined as follows:

$$\forall t[\mathsf{G}(t, \text{GOTO}(r,p)) \rightarrow \mathsf{B}(t, \partial^2 pos(r) := 0.5 * derdir(attid(r)))]$$

where the auxiliary function *derdir* is defined as

$$\{\mathsf{N} \mapsto (0,1), \mathsf{S} \mapsto (0,-1), \mathsf{E} \mapsto (1,0), \mathsf{W} \mapsto (-1,0)\}.$$

(Unfortunately it does not come out the same as *cdir*). Here, the operator ∂ represents differentiation on both components of the position, ∂^2 is the second derivative, so it results in the acceleration viewed as a vector. No rotation is involved. Notice that we specify the robot's velocity in floorplan coordinates, and not relative to the direction where it is "looking". This is because we wish to treat the floorplan-coordinate position as the primary one, which is subject to the common routines about restriction of change, whereas robot coordinates must always be derived. (When the robot moves, then everything else in the room changes its position in the robot coordinates, so this does not go well with an assumption of more or less strict inertia or persistence).

Notice also that this definition specifies the new value (by the assignment operation) of *believed* acceleration. We use this here for indicating the value that is given to the actuator. Actually, it might have been preferable to have a third relation R besides H and B, reserving B for estimated feature values, and using R for reference values that the controller is requested to maintain. However, for the purpose of the present article it is sufficient to identify R with B, and to assume that the invocation of the action sets the appropriate value for the robot's acceleration for an instant, but for all subsequent times the believed value of the acceleration will be the estimated one.

We also stipulated an assumption that an initial acceleration of 0.5 will be held until the robot reaches the velocity of 1 m/s, and then the speed will become constant. We model this as change as something which happens without explicit intervention by the ego, and which therefore has the same status as a law of nature. It is expressed by

$$|\partial pos[r]\,| = 1 \wedge \ |\partial^2 pos[r]\,| \neq 0 \rightarrow |\partial^2 pos[r]\,| := 0$$

This applies of course for an idealized world; in a more realistic case one would have to compensate to hold velocity constant at the value 1.

Finally, we have to specify that the elementary action terminates with success if the robot reaches the target position:

$$\mathsf{B}(t, yc(mc(p,r)) = 0) \wedge \mathsf{D_c}([s,t], \textsc{goto}(r,p)) \rightarrow$$

$$\mathsf{D_s}([s,t], \textsc{goto}(r,p)) \ \wedge \mathsf{B}(t, \partial pos(r) := (0,0)) \wedge$$

$$\mathsf{B}(t, \partial^2 pos(r) := (0,0))$$

4.4 Failure conditions for the action

We arrive now to the specifications of how the action can fail. In general, for each action there is a multitude of reasons why it may fail: various things can break, obstacles may come into the way, etc etc. In the AI literature this has been described by the term *qualification problem* (although the meaning of that term seems to have slipped in the last few years). It has been recognized since very long [3] that a nonmonotonic logic is needed for this: it is not possible to enumerate all failure reasons at once; some failures may be implied indirectly from various other and more general circumstances, and we wish to say that the action succeeds unless there is *some* reason why it fails.

The present logical system is no exception; the nonmonotonicity is introduced by preferring a model where an action continues over a model where it terminates, either by success or by failure.

To show how this works, consider the case where the ego terminates the action as failed if its proximity sensor $obstac(r)$ senses something in the way although the destination object has not been reached. The corresponding proposition is as follows.

$$\mathsf{B}(t, obstac(r)) \wedge \mathsf{D}_{\mathrm{n}}([s,t], \textsc{goto}(r,p)) \rightarrow$$
$$\mathsf{D}_{\mathrm{f}}([s,t], \textsc{goto}(r,p)) \;\wedge \mathsf{B}(t, \partial pos(r) := (0,0)) \wedge$$
$$\mathsf{B}(t, \partial^2 pos(r) := (0,0))$$

Notice how this rule simply implies a D_{f} expression for the action; other axioms may imply failure termination independently using similar consequents. The second condition for interrupting the action, namely that someone pushed a emergency button, can be added as follows:

$$\mathsf{B}(t, emergency) \wedge \mathsf{D}_{\mathrm{n}}([s,t], \textsc{goto}(r,p)) \rightarrow$$
$$\mathsf{D}_{\mathrm{f}}([s,t], \textsc{goto}(r,p)) \;\wedge \mathsf{B}(t, \partial pos(r) := (0,0)) \wedge$$
$$\mathsf{B}(t, \partial^2 pos(r) := (0,0))$$

Referring back to the general axioms for the variants of the D operator, we see how the action, once invoked, is able to continue towards a success (when the successful termination condition is finally satisfied) unless there is a qualification that forces it to fail.

With this technique, additional reasons for failure of the action can be added in a modular fashion.

5 Nonmonotonicity

The first-order formalism using the *Holds* and *Do* predicates allows a considerable flexibility in expressing actions invoked by one or more agents, events which are caused directly or indirectly by the actions of agents, or which simply arise due to the laws of nature, extended duration and concurrency of actions and events, and so on. The present paper has scetched how this formalism can also deal with hybrid (mixed continuous-discrete) systems and with failure conditions of actions.

One very characteristic feature of the formalism that is obtained in this fashion is that, although it is based on classical FOPC, it must also incorporate some *nonmonotonic* technique in order that the intended conclusions shall be obtained for given scenarios. Monotonicity, in this context, means that if S is a set of axioms, $Th(S)$ is the set of conclusions that follow from S, and $S \subseteq S'$, it follows that $Th(S) \subseteq Th(S')$. In other words, previously obtained conclusions can be retained when more axioms are added. It is this traditional monotonicity property that is *not* upheld in logics of actions and change.

Technically speaking, nonmonotonicity is obtained by using *entailment methods* which restrict the set of models. If S is a set of axioms and $Mod(S)$ is the set of its classical models, then an entailment method is a way of obtaining a subset of $Mod(S)$ called the set of *selected models*. The corresponding set $T(S)$ of nonmonotonic conclusions is the set of formulae true in all selected models, so it is a superset of $Th(S)$. A number of entailment methods have been proposed and analyzed; most of them are based on the notion of imposing a preference relation on models, and of selecting models which are maximal with respect to the preference relation.

The traditional reason for introducing nonmonotoniciy in logics of actions and change was in order to capture intended closure properties with respect to the set of actions and to their effects. One wanted to assume that the actions that were explicitly stated in the scenario axioms were *the only relevant actions* in that scenario. One also wanted to assume that the effects of actions that were stated in an action law were *the only relevant effects* of the action described in the action law. Finally, it was recognized in general terms that qualification requires nonmonotonicity.

These rationale for nonmonotonicity continue to be valid, but the context of hybrid scenarios and actions for which a success/fail distinction has been defined add more to the picture. In particular, it allows us to bring a concrete formal treatment to the traditional intuitions about qualification.

Let us illustrate these aspects briefly through an example. Suppose one first describes a scenario through the following statements: at time 0 the robot r_4 is standing still at the origo, looking north, it is 2 units wide and 0.5 unit long, and there is no obstacle and noone pushes the emergency button. No action is performed. Furthermore, the effects of robots actions are as was specified above.

It is straight-forward to express these statements in the notation described here. The statement that no action is performed is expressed simply by not making any statement using the predicates D or G.

Under the assumptions mentioned above, one can conclude that at time 20, the robot is still in the position (0,0). Consider now the effect of adding the statement that at time 9, the robot r_4 starts to go towards position (0.5, 10), that is, one adds

$$\mathsf{G}(9, \textsc{goto}(r_4, (0.5, 10)))$$

to the set of axioms. The logic should then be able to conclude that at time 20 the robot is in position (0,10), and that the GOTO action has been executed successfully over the time interval (9,20). More precisely, the given set of axioms after the addition of the invocation statement should have only one model, namely the one where the robot moves according to the given laws, and where

$$\mathsf{D}_s([9, 20], \textsc{goto}(r_4, (0.5, 10)))$$

is entailed. Notice that the previous conclusion concerning the robot's position at time 20 is now invalid, and that another conclusions has taken its place. Notice also that the conclusion presently obtained depends crucially on the fact that no obstacle and no emergency need to be assumed during the interval (9,20).

Next, if one adds the additional statement

$$\mathsf{H}(14, \mathit{emergency})$$

which says that the emergency button was pushed at time 14, then the previous conclusion that the goto action succeeded over the interval $[9, 20]$ shall be replaced by the conclusion that it failed over the interval $[9, 14]$. The same holds, of course, if the statement

$$\mathsf{H}(14, obstac(r_4))$$

is instead added as an axiom.

We see, therefore, that the characterization of this kind of hybrid system using a logic of action and change truly requires the logic *both* to characterize actions in terms of their operational definition, *and* to characterize the piecewise continuous but occasionally discontinuous behavior of positions, velocities, accelerations, etc. In this context, the change of value of a discrete-valued state variable is simply a special case of the discontinuity of a continuous-valued variable.

We described in some earlier papers [5, 4] how this type of hybrid change can be accomodated in a first-order logic. This approach is based on a few basic concepts:

- The basis of the approach is to embed differential calculus within a multi-sorted first-order logic with metric time as one of the sorts.
- Nonmonotonicity is dealt with by generalizing the traditional concept of minimizing or restricting change of discrete-valued fluents, to similarly minimizing or restricting discontinuities in piecewise continuous fluents [5, 4, 11].
- An additional key technique is to use the *occlusion* operator X that was introduced by us in [6] (it was later renamed "release" by Kartha and Lifschitz in [1]). We write $\mathsf{X}(t, f)$ for saying that the fluent f is exempt from the restriction against discontinuities that applies elsewhere along the timeline. The possible range of values for occluded features is specified by action laws or other axioms.
- The reassignment operator := is used as an abbreviation: $\mathsf{H}(t, f := v)$ is defined to mean $\mathsf{H}(t, f : v) \wedge \mathsf{X}(t, f)$, and similarly for B.

For the kind of robotic scenario that is discussed in the present article, one can use these principles – minimization or restriction of discontinuities, use of an occlusion operator, and viewing assignment as an abbreviation – both for the interpretation of the operational definitions of actions, and for describing the "physics" of the world where the robots operate.

One important feature of this approach is that it provides a unified way of interpreting the detailed behavior of the robot and the physical laws in the robot's environment. This means, for example, that the effect obtained by adding an axiom which explicitly says there is an obstacle at time 14, will also be obtained if one adds statements involving an additional robot r_2 whose movements are defined so that it gets in the way of r_4 at time 14. Thus, reasoning about qualifications or failure conditions is completely integrated into this nonmonotonic logic.

6 Relating the effect definition and the operational definition

We have now developed the effect definition and the operational definition side by side; let us denote them Σ_E and Σ_P (for program), respectively. The problem that we chose initially was how one can verify that Σ_E follows from Σ_P, using also the other relevant axioms (axioms for the D operators, definitions of how aggregated features depend on basic ones, definitions of sensors, etc).

At first sight, this appears to be straightforward, both in an informal sense and a formal sense. Informally, in any model where $\mathsf{D}_\mathsf{s}([s,t], \text{GOTO}(m))$ is true and all the axioms stated above are satisfied, the following must be the case:

1. Applicability holds (otherwise the action would have failed)
2. Invocation has been performed
3. The physical laws apply
4. No obstruction along the execution (because if so the action would have failed)
5. Condition for successful termination must be satisfied
6. Termination operations (resetting velocity to zero, etc) must have been performed.

From this, and also using the assumptions about sensor correctness, it should be straightforward to conclude the success description, and (with a bit more effort) the failure description of the action.

Similarly on the formal side: all the formulae are expressed in the same logical language; it is a first-order logic although a somewhat complicated one; its semantics is well defined. Isn't it just a matter of showing that the set of models of Σ_E contains the models for Σ_P?

This classical perspective has to be modified because of the need for non-monotonicity. The correctness criterion becomes, instead, that the set of *selected* models for Σ_E contains the selected models for Σ_P.

Entailment methods for actions which are described on the discrete level have been studied extensively, and relatively much is known about their range of applicability and other similar properties [7, 8, 9]. The effect-description level is simpler to deal with in the sense that one can (often) restrict it to only using discrete-valued state variables, but it also suffers from an additional complication: modelling the world in terms of qualitative change gives rise to apparent nondeterminism in addition to the one which may be genuine in the application. Entailment methods for hybrid systems involving mixed continuous and discrete change have been less studied, and no formal results about their range of correct applicability have been obtained yet. This is therefore a topic of continued research.

7 Composition of failible actions

The operative definitions of actions will usually be constructed by composition of simpler actions. It is straight-forward, in the approach described here, to introduce the customary operators for composition of actions, such as sequential composition, conditional choice of action, iteration of actions, successive attempts of several actions until one of them succeeds, and so forth. We have shown (see the WWW page mentioned at the end of the article) how this can be done, and that the required additional axioms are quite simple and straightforward. In fact, the verification criterion for low-level actions can be directly applied to high-level actions as well. This is because both kinds of actions as well as the composition operators have been defined in essentially the same logic.

Notice, by the way, that the operational definitions described here represent a kind of "programming language" for the actions; they describe the actions as seen from the executing robot, and not as seen from "outside". It is therefore quite natural to construct higher-level operations using composition operators that are familiar from programming languages.

8 Summary

We addressed the problem of formally proving high-level effect descriptions of actions from low-level operational definitions. Both descriptions are expressed in a logic of actions and change, with extensions for characterizing continuous change, discontinuities, the distinction between true and estimated values of state variables, and the distinction between success and failure of an action. Both descriptions also require the use of nonmonotonicity in the logic in question. The transition from operational definition to effect description furthermore involves the creation of a closure with respect to the set of possible ways that the action can fail.

We have outlined, by means of an example, how these issues can be addressed as an extension of existing results on logics of actions and change.

Additional material

The present paper is fairly brief due to the space constraints of the present workshop proceedings. The following WWW page contains links to additional articles on the same subject:

`http://www.ida.liu.se/~erisa/valrok/`

and it will be maintained for the foreseeable future. A longer article has been published electronically by Linköping University Electronic Press, and will be maintained permanently at:

`http://www.ep.liu.se/ea/cis/1996/004/`

It contains some of the material that has been mentioned above as missing in the present article (axioms omitted in subsection 2.3, axioms for action composition operators in section 7, etc).

References

1. G. Neelakantan Kartha and Vladimir Lifschitz. Actions with indirect effects (preliminary report). In *International Conference on Knowledge Representation and Reasoning*, pages 341–350, 1994.
2. Vladimir Lifschitz. Nested abnormality theories. *Artificial Intelligence*, 74:351–365, 1995.

3. John McCarthy and Patrick Hayes. Some philosophical problems from the viewpoint of artificial intelligence. In *Machine Intelligence, Vol. 4*, pages 463–502. Edinburgh University Press, 1969.
4. Tommy Persson and Lennart Staflin. A causation theory for a logic of continuous change. In *European Conference on Artificial Intelligence*, pages 497–502, 1990.
5. Erik Sandewall. Combining logic and differential equations for describing real-world systems. In *Proc. International Conference on Knowledge Representation, Toronto, Canada*, 1989.
6. Erik Sandewall. Filter preferential entailment for the logic of action in almost continuous worlds. In *International Joint Conference on Artificial Intelligence*, pages 894–899, 1989.
7. Erik Sandewall. The range of applicability of nonmonotonic logics for the inertia problem. In *International Joint Conference on Artificial Intelligence*, 1993.
8. Erik Sandewall. *Features and Fluents. The Representation of Knowledge about Dynamical Systems. Volume I.* Oxford University Press, 1994.
9. Erik Sandewall. Assessments of ramification methods that use static domain constraints. In *International Conference on Knowledge Representation and Reasoning*. Morgan Kaufmann, 1996.
10. Erik Sandewall and Ralph Rönnquist. A representation of action structures. In *National Conference on Artificial Intelligence*, pages 89–97, 1986.
11. Murray Shanahan. Representing continuous change in the event calculus. In *European Conference on Artificial Intelligence*, pages 598–603, 1990.
12. Yoav Shoham. *Reasoning about Change.* MIT Press, 1988.
13. Michael Thielscher. Computing ramifications by postprocessing. In *International Joint Conference on Artificial Intelligence*, 1995.

This article was processed using the LaTeX macro package with LLNCS style