Reasoning by Regression: Pre- and Postdiction Procedures for Logics of Action and Change with Nondeterminism*

Marcus Bjäreland and **Lars Karlsson**
Department of Computer and Information Science
Linköpings University, S-581 83 Linköping, Sweden
email: {marbj, larka}@ ida.liu.se

Abstract

In this paper we introduce regression-based pre- and postdiction procedures for PMON, a nonmonotonic logic for action and change with explicit time. We also provide an in depth analysis of problems with regression that occur when nondeterministic actions are introduced. We do this by employing Dijkstra's weakest liberal precondition operator, *wlp*. The presented work is related to recent work by Lin in situation calculus, and we identify and deal with three problems with his approach. Our conjecture is that our approach can be mapped back to situation calculus.

1 Introduction

The contributions of this paper are the following: Regression-based pre- and postdiction procedures for PMON with metric time, and an in depth analysis of problems with regression that occur when nondeterministic actions are introduced. We also identify and deal with problems in previous work by Lin [1996].

Lin [1996] proposes a causal minimization framework in SitCalc (Situation Calculus), where effects of nondeterministic actions can be specified. He then identifies a class of such actions for which regression can be used as a reasoning mechanism. Within the constraints of the SitCalc formalism, Lin provides an elegant extension to handle nondeterminism, but one can probably do somewhat better. We will show this, using PMON as a vehicle.

PMON is a logic for action and change proposed by Sandewall [1994]. PMON is proven to be correctly applicable to problems involving explicit time, context-dependent actions, nondeterministic actions and actions with duration. Doherty [1994] gives a first-order formulation of PMON, with a second-order circumscription axiom and shows that it is always possible to reduce the second-order theory to first-order logic. In [Gustafsson and Doherty, 1996] PMON is further extended to handle causal constraints. In this paper we will use a subset of PMON with metric time (natural numbers) and propositional fluents.

There are three problems with Lin's proposal: First, he needs to explicitly enumerate all nondeterministic effects of an action in syntax, thus encountering a possible combinatorial explosion. Secondly, he has to introduce an "oracle" predicate *Case*, which "knows" the outcome of a nondeterministic action before it has occurred. The intuitive meaning of this oracle is unclear. Thirdly, the effects of nondeterministic actions are not allowed to depend on the situation in which the action is invoked, for regression to work properly.

In PMON, we deal with each of the three problems in the following manner: The first is taken care of by the formalism, since nondeterministic effects can be represented succinctly. For the the second and third problem we rely on our the minimization of change policy, which only allows an action to have certain effects if the corresponding condition is satisfied. This behavior is similar to the behavior of conditional statements in programming languages, which enables us to use a classical computer science tool, Dijkstra's *weakest liberal precondition* operator, *wlp*, for regression, which, as it turns out, deals with the problems.

The comparison between our and and Lin's approach will be informal due to the use of different time structures. We will, however, let Lin's work guide the development of the results presented in this paper.

2 Goal Regression in SitCalc and Lin's Proposal

Initially, goal regression was developed as a plan synthesis method ([Waldinger, 1977], [Pednault, 1986]). Reiter [1991] provided a situation calculus account for goal regression with a completeness proof. Reiter relies on his solution to the frame problem, in particular the successor state axioms, which are biconditional relations between the values of a fluent *after* the execution of an action, and the states *before* the action. The general form of these axioms, for every fluent R, is

$$\forall a, s. Poss(a, s) \rightarrow [h(R, do(a, s)) \equiv$$

*This research has been supported by the Swedish Research Council for Engineering Sciences (TFR) grants 93-183 and 96-728.

$$\gamma_R^+(a,s) \vee h(R,s) \wedge \neg\gamma_R^-(a,s)],$$

where $Poss(a,s)$ is true if the action a is possible to execute in situation s, and $h(R,s)$ denotes that the fluent R is true in situation s. The formula $\gamma_R^+(a,s)$ denotes the conditions under which action a will make R true in situation s, and $\gamma_R^-(a,s)$ the conditions that will make R false.

By substituting fluents in the goal with the right hand side of the biconditional in the successor state axioms, the nesting of the *do* function can be reduced until there finally is a formula only mentioning the situation S_0, on which a classical atemporal theorem prover can be used. For planning purposes, Reiter's approach relies on completely specified initial states, and cannot handle nondeterministic actions.

Lin introduces a predicate *Caused* which "assigns" truth values to fluents. In the minimization policy, the extension of *Caused* is minimized and a no change premise is added to the theory. To illustrate his approach Lin uses the dropping-a-pin-on-a-checkerboard example: There are three fluents, *white* (to denote that the pin is partially or completely within a white square), *black*, and *holding* (to denote that the pin is not on the checkerboard). There are two actions, *drop* (the pin is drop onto the checkerboard) and *pickup*. In this paper we will only consider the *drop* action (which is the only one with nondeterministic effects), which is formalized as follows:

$$\begin{aligned}
&\forall s.Poss(drop,s) \rightarrow \\
&\quad [Caused(white,true,do(drop,s)) \wedge \\
&\quad Caused(black,true,do(drop,s))] \vee \\
&\quad [Caused(white,false,do(drop,s)) \wedge \\
&\quad Caused(black,true,do(drop,s))] \vee \\
&\quad [Caused(white,true,do(drop,s)) \wedge \\
&\quad Caused(black,false,do(drop,s))]. \qquad (1)
\end{aligned}$$

The *Poss* predicate is defined, with an action precondition axiom, as

$$\begin{aligned}
&\forall s.Poss(drop,s) \equiv \\
&\quad h(holding,s) \wedge \neg h(white,s) \wedge \neg h(black,s)
\end{aligned}$$

A problem with this is that the effects of the action have to be explicitly enumerated in the action definition. The number of disjuncts will grow exponentially in the number of fluents, and may become problematic in implementations.

3 PMON

In PMON it is possible to model nondeterminism in a more succinct manner. We will present the parts of PMON relevant for this paper. Extensions and details can be found in [Doherty, 1996].

We will use the language $\mathcal{L}$ which is a many-sorted first-order language. For the purpose of this paper we assume two sorts: A sort $\mathcal{T}$ for time and a sort $\mathcal{F}$ for propositional fluents. The sort $\mathcal{T}$ will be the set of natural numbers.

The language include two predicate symbols H[1] and *Occlude*, both of type $\mathcal{T} \times \mathcal{F}$. The numerals $0, 1, 2, \ldots$ and the symbols $\mathbf{s}$ and $\mathbf{t}$, possibly subscripted, denote natural numbers, i.e. constants of type $\mathcal{T}$ (*time points*).

PMON is based on *action scenario descriptions* (scenarios), which are narratives with three components: OBS, a set of observations stating the values of fluents at particular time points, ACT, a set of action laws, and SCD, a set of schedule statements that states which and when (in time) actions occur in the scenario.

Example 3.1 A scenario where the pin initially was held over the checkerboard, then dropped and finally observed to be on a white square (and not on a black) is formalized as

OBS1	$H(0, over\_board) \wedge \neg H(0, white) \wedge \neg H(0, black)$
ACT	$[s,t]Drop \Rrightarrow H(s, over\_board) \rightarrow [s,t]\gamma$
SCD	$[3,5]Drop$
OBS2	$H(6, white) \wedge \neg H(6, black)$,

where $[s,t]\gamma$ is the formula

$$\begin{aligned}
&[s,t]over_board{:=}\mathbf{F} \wedge \\
&(([s,t]white{:=}\mathbf{T} \wedge [s,t]black{:=}\mathbf{T}) \vee \\
&([s,t]white{:=}\mathbf{F} \wedge [s,t]black{:=}\mathbf{T}) \vee \\
&([s,t]white{:=}\mathbf{T} \wedge [s,t]black{:=}\mathbf{F})). \qquad (2)
\end{aligned}$$

□

Below, in (3), we will show how $[s,t]\gamma$ can be presented in a succinct way.

In Lin's formalism $h(holding, s)$ is a *qualification* of the action *drop*. In PMON, such qualifications cannot (yet) be modeled. To illustrate conditions for the action to have effects, we introduce *over_board*, that denotes that *Drop* only has effects if the pin is dropped on the board. Such conditions can be modelled in SitCalc, but not in Lin's framework. This will be further discussed below.

H formulae and Observations

Boolean combinations of H literals (i.e. possibly negated atomic formulae), where every literal mentions the same time point, are called *H formulae*. An H formula, δ, where every literal only mentions time point $\mathbf{s}$, will sometimes be written $[\mathbf{s}]\delta$. An *observation* is an H formulae.

Action Laws

A *reassignment* is a statement $[\mathbf{s},\mathbf{t}]f{:=}b$, where $f \in \mathcal{F}$ and b is a truth value symbol ($\mathbf{T}$ for true, and $\mathbf{F}$ for false). The statement $[\mathbf{s},\mathbf{t}]f{:=}\mathbf{T}$ is an abbreviation of $H(\mathbf{t},f) \wedge \forall t.\mathbf{s} < t \leq \mathbf{t} \rightarrow Occlude(t,f)$, and $[\mathbf{s},\mathbf{t}]f{:=}\mathbf{F}$ expands to $\neg H(\mathbf{t},f) \wedge \forall t.\mathbf{s} < t \leq \mathbf{t} \rightarrow Occlude(t,f)$. A *reassignment formula* $[\mathbf{s},\mathbf{t}]\alpha$, is a boolean combination of reassignments, all mentioning the same temporal constants $\mathbf{s}$ and $\mathbf{t}$.

An *action law* is a statement

$$\begin{aligned}
&[s,t]A \Rrightarrow \\
&\quad \textstyle\bigwedge_{i=1}^{n}[[s]precond_i^A \rightarrow [s,t]postcond_i^A],
\end{aligned}$$

[1] Not to be confused with h which is the corresponding situation calculus predicate.

where A is an action designator (the name of the action), s and t are variables (*temporal* variables) over the natural numbers, each $precond_i^A$ is either a conjunction of H literals only mentioning the temporal variable s, or the symbol $\mathbf{T}$, such that the $precond_i^A$'s are mutually exclusive, and every $postcond_i^A$ is a reassignment formula. Intuitively, ACT contains *rules* for expanding schedule statements into action axioms, in which the temporal variables are instantiated.

An action is said to be *deterministic* iff none of its postconditions include disjunctions. An action is *admissible* if no precondition or postcondition is a contradiction. We will call the conjuncts of an action law *branches.*

It is necessary that the reassignments in the postconditions are on a form logically equivalent to *full trajectory normal form* (FTNF), i.e. a special disjunctive normal form, where each disjunct contains the same fluents, e.g. for the formula (2) above. Of course, this looks suspiciously similar to Lin's formulation, which was criticised above. But this formula is an abbreviation of a formula equivalent to

$$\begin{aligned}&\neg H(t, over\_board) \wedge (H(t, white) \vee H(t, black)) \wedge\\ &\forall t'.s < t' \leq t \rightarrow Occlude(t', holding) \wedge\\ &Occlude(t', white) \wedge Occlude(t', black), \qquad (3)\end{aligned}$$

which deals with the problem of compact and intuitive representation. We will use the abbreviated form for readability reasons, but all the constructions in this paper could easily be redone for action effects of the type in (3).

Schedule Statements

A *schedule statement* is a statement, $[\mathbf{s}, \mathbf{t}]A$, such that $\mathbf{s}$ and $\mathbf{t}$ are natural numbers, $\mathbf{s} < \mathbf{t}$, and A is an action designator for which there exists an action law in ACT.

No Change Premises

The occlusion of fluents that are reassigned captures the intuition of possible change, but we also need a syntactic component that captures the intuition that a fluent cannot change value *unless* it is occluded. This is taken care of by the *no change premise*, NCH, formulated as

$$\forall f, t.\neg Occlude(t+1, f) \rightarrow (H(t, f) \equiv H(t+1, f)).$$

Action Scenario Description

An *action scenario description* (scenario) is a triple $\langle \text{OBS}, \text{ACT}, \text{SCD} \rangle$, where OBS is a set of observations, ACT a set of action laws, and SCD a set of schedule statements. Furthermore, let SCD(ACT) denote a set of formulae such that we for every schedule statement in SCD instantiate the temporal variables of the corresponding action law (in ACT) with the two time points in the schedule statement. We will call such formulae *action instances.*

PMON Circumscription Policy

To minimize change, the action instances are circumscribed (by second-order circumscription), i.e. by minimizing the *Occlude* predicate, and then by filtering with the observations and no change premises . For a scenario description $\Upsilon = \langle \text{OBS}, \text{ACT}, \text{SCD} \rangle$ we have

$$\begin{aligned}&PMON(\Upsilon) =\\ &\quad Circ_{SO}(\text{SCD}(\text{ACT})(Occlude); \langle Occlude \rangle) \cup\\ &\quad \{\text{NCH}\} \cup \text{OBS} \cup \Gamma,\end{aligned}$$

where Γ is the set of unique names axioms. Doherty [1994] shows that this second-order theory always can be reduced to a first-order theory that provide a *completion* (or, definition) of the *Occlude* predicate. For the needle dropping scenario the completion will be the following:

$$\forall t, f.Occlude(t, f) \equiv \bigvee [t = t' \wedge f = f'],$$

for every combination of $t' \in \{4, 5\}$ and $f' \in \{over\_board, white, black\}$.

A classical model for a circumscribed scenario will be called an *intended model.* A scenario is consistent if it has an intended model, else it is inconsistent. We say that a scenario Υ *entails* a statement $[\mathbf{t}]\varphi$, and writes $\Upsilon \mathrel{|\!\!\sim} [\mathbf{t}]\varphi$, iff $PMON(\Upsilon) \vdash [\mathbf{t}]\varphi$, where $\vdash$ denotes classical deductive inference. Furthermore, a schedule statement $[\mathbf{s}, \mathbf{t}]A$ is said to entail a statement $[\mathbf{t}]\varphi$, written $[\mathbf{s}, \mathbf{t}]A \mathrel{|\!\!\sim} [\mathbf{t}]\varphi$, iff $\langle \emptyset, \text{ACT}, [\mathbf{s}, \mathbf{t}]A \rangle \mathrel{|\!\!\sim} [\mathbf{t}]\varphi$.

The completion of *Occlude* has a useful ramification: It *conditionalizes* the branches of the actions in the scenario, i.e. when the completion axiom is added to the theory, it is no longer possible for a precondition to be false while its postconditions are true. This behavior is similar to the behavior of if-then statements in programming languages, and makes it possible to use Dijkstra's *wlp* formula transformer.

4 Lin's Proposal (Cont'd)

To be able to use goal regression, Lin has to generate successor state axioms. However, this cannot be done in a straightforward way for nondeterministic scenarios. The reason for this is that there are no constraints *before* a nondeterministic action in biconditional relation with what holds *after* the action has taken effect.

Lin deals with this by introducing an "oracle[2]" predicate, $Case(n, a, s)$, which is true iff the nth disjunct of action a has effect (is true) in situation s. The *drop* action is, thus, defined as

$$\begin{aligned}&\forall a, s.Poss(drop, s) \wedge Case(1, drop, s) \rightarrow\\ &\quad Caused(white, true, do(drop, s)) \wedge\\ &\quad Caused(black, true, do(drop, s)),\\ &\forall a, s.Poss(drop, s) \wedge Case(2, drop, s) \rightarrow \ldots\end{aligned}$$

Lin shows that the new theory is a conservative extension of the previous circumscribed one.

The key here is that a previously nondeterministic action is transformed into a deterministic action, where nondeterminism is simulated by the oracles. Lin suggest that the oracles could be viewed as probabilities

[2] A term used by Sandewall for constructs that "know" in advance what the effect of nondeterministic actions will be.

of certain effects to take place. However, since he does not develop this idea, the introduction of oracles is not completely convincing, at least not from a knowledge representation point of view. We will, below, present a vehicle to avoid the use of oracles.

Since the actions now are deterministic it is possible to generate successor state axioms, e.g. for *white*:

$$\forall a, s.Poss(a,s) \rightarrow [h(white, do(a,s)) \equiv \\ a = drop \wedge (Case(1, drop, s) \vee Case(3, drop, s))]$$

Unfortunately, this approach is not as general as it may seem. In fact, goal regression is not possible unless we restrict the problem by disallowing the effects of nondeterministic action to be conditional. We cannot, e.g., introduce the *over_board* condition on the effects of the *drop* action.

5 Dijkstra's Semantics Applied to PMON

Since we cannot generate the desired biconditionals, we would like to automatically generate the "best" possible condition before the action. Also, such an approach should be able to handle conditional nondeterministic actions, to deal with this particular problem of Lin's proposal. We achieve the result by using a classical computer science approach, the weakest liberal precondition operator, *wlp*.

In this section we briefly introduce Dijkstra's semantics for programming languages, but in the setting of PMON. We also present theorems which connect the states before and after the execution of actions. Originally, the theory was developed to give semantics to small programming languages with a higher ordered function. This function was a predicate transformer, that given a command, S, in the language and a machine state, s_i, returned a set, W, of machine states such that all states in which S could be executed and terminate in s_i, if it at all terminated, belonged to W. This predicate transformer was called *weakest liberal precondition*, *wlp*.

Dijkstra's semantic has previously been used in an action and change setting in [Lukaszewicz and Madalinska-Bugaj, 1995]. The language used in there is a small programming language, without a notion of explicit time. They does not specifically deal with regression.

Let φ be a H formula, f a fluent and $b \in \{\mathbf{T}, \mathbf{F}\}$. We write $[\mathbf{s}]\varphi[f \leftarrow b]$ to denote that all H statements (Note: *Not* the possible negation in front of them) mentioning the fluent f are simultaneously replaced by b, and where the time argument of the H statements are change to $\mathbf{s}$. As a consequence, $[\mathbf{s}]\varphi[]$ denotes the replacement of all time arguments in φ with $\mathbf{s}$.

We will informally use the term *state* for a propositional interpretation of all fluents at a specific time point.

Definition 5.1 (Weakest liberal precondition, *wlp*)
Let α and β be reassignment formulae, φ a H formula, we define *wlp* inductively so that, for all fluents $f \in \mathcal{F}$

$$wlp([\mathbf{s},\mathbf{t}]f{:=}\mathbf{T}, \varphi) \stackrel{\text{def}}{=} [\mathbf{s}]\varphi[f \leftarrow \mathbf{T}] \tag{4}$$

$$wlp([\mathbf{s},\mathbf{t}]f{:=}\mathbf{F}, \varphi) \stackrel{\text{def}}{=} [\mathbf{s}]\varphi[f \leftarrow \mathbf{F}] \tag{5}$$

$$wlp(\alpha \wedge \beta, \varphi) \stackrel{\text{def}}{=} wlp(\beta, wlp(\alpha, \varphi)) \tag{6}$$

$$wlp(\alpha \vee \beta, \varphi) \stackrel{\text{def}}{=} wlp(\alpha, \varphi) \wedge wlp(\beta, \varphi) \tag{7}$$

Furthermore, let $[\mathbf{s},\mathbf{t}]A$ be a schedule statement for which the corresponding law has n branches. Then

$$\begin{aligned} & wlp([\mathbf{s},\mathbf{t}]A, [\mathbf{t}]\varphi) \stackrel{\text{def}}{=} \\ & \quad (\textstyle\bigwedge_{i=1}^{n}[precond_i^A \rightarrow wlp(postcond_i^A, \varphi)]) \\ & \qquad \wedge \\ & \qquad (\textstyle\bigwedge_{i=1}^{n}[\neg precond_i^A] \rightarrow [\mathbf{s}]\varphi[]) \end{aligned} \tag{8}$$

The second conjunct encodes that if none of the preconditions were true then φ had to be true at the beginning of the action. We define *wlp*'s *conjugate*, wlp^*, as $wlp^*(S, \alpha) = \neg wlp(S, \neg\alpha)$.□

Example 5.2 We briefly illustrate *wlp* by computing the weakest liberal precondition for *white* or *black* to hold after the *Drop* action (from example 3.1). Obviously, the only way in which we can be guaranteed that *white* or *black* is true after *Drop* is if *over_board*, or *white* or *black*, was true before the action.

$$\begin{aligned} & wlp([3,5]Drop, H(5, white) \vee H(5, black)) = \\ & \quad (H(3, over\_board) \rightarrow \\ & \quad wlp([3,5]\gamma, H(5, white) \vee H(5, black)) \wedge \\ & \quad (\neg H(3, over\_board) \rightarrow \\ & \quad H(3, white) \vee H(3, black)). \end{aligned} \tag{9}$$

Now, we compute $wlp([3,5]\gamma, H(5, white) \vee H(5, black))$. The first conjunct of $[3,5]\gamma$, $[3,5]over\_board{:=}\mathbf{F}$, will not have any effect on $H(5, white) \vee H(5, black)$. When *wlp* is applied to the three disjuncts of $[3,5]\gamma$ they will all be computed to $\mathbf{T}$. Thus, $wlp([3,5]\gamma, H(5, white) \vee H(5, black)) = \mathbf{T}$, and (9) will be equivalent to

$$\begin{aligned} & (H(3, over\_board) \rightarrow \mathbf{T}) \wedge \\ & (\neg H(3, over\_board) \rightarrow H(3, white) \vee H(3, black)) \\ & \equiv \\ & \neg H(3, over\_board) \rightarrow H(3, white) \vee H(3, black), \end{aligned}$$

which is what we wanted.□

Note that *wlp* is applied to the possible *effects* of an action in (4) – (7), and is generalized to complete action instances in (8).

The correctness of *wlp* is proven in [Dijkstra and Scholten, 1990]. We will, however, give intuitions for definitions (6) – (8).

Definition (6) is based on a sequential underlying computational model. If we implement *wlp* we will have to perform the reassignments in some order, and here we choose to first apply α and then β.

Definition (7) handles nondeterminism. It guarantees that no matter which of α or β is executed, we will end up in a state satisfying φ. The only states that can guarantee this are those that can start in α *and* β and end in φ.

For (8) we can note that *wlp* should generate every state, σ, such that if a precondition is true in σ, *wlp* applied to the corresponding postcondition and some φ should be true in σ too. This excludes every action branch that would not terminate in a state satisfying φ. The second conjunct of (8) makes *wlp* work exhaustively, i.e. if none of the preconditions are true then φ was true before the action.

The conjugate $wlp^*([\mathbf{s},\mathbf{t}]A, [\mathbf{t}]\varphi)$ should, analogously to *wlp*, be interpreted as: "The set of all states at $\mathbf{s}$ such that the execution of A in any of them does *not* end in a state at $\mathbf{t}$ satisfying $\neg\varphi$."

For all the theorems below we assume that the actions are admissible.

Proposition 5.3 $[\mathbf{s},\mathbf{t}]A \mathrel{|\!\sim} wlp([\mathbf{s},\mathbf{t}]A, [\mathbf{t}]\varphi) \rightarrow [\mathbf{t}]\varphi$.
Proof (sketch): Follows from the correctness of *wlp*.□

Corollary 5.4 $[\mathbf{s},\mathbf{t}]A \mathrel{|\!\sim} [\mathbf{t}]\varphi \rightarrow wlp^*([\mathbf{s},\mathbf{t}]A, [\mathbf{t}]\varphi)$.
Proof: Contraposition of the implication yield a formula to which proposition 5.3 is applicable.□

Proposition 5.5 $wlp(S,\varphi) \equiv wlp^*(S,\varphi)$ hold for deterministic actions S.
Proof (sketch): Structural induction over the parts of *wlp*'s definition. For the application of *wlp* to actions, we heavily rely on the fact that the preconditions are mutually exclusive, and that *wlp* adds exhaustiveness.□

Proposition 5.6 Let $[\mathbf{s},\mathbf{t}]A$ be a deterministic action. Then

$$[\mathbf{s},\mathbf{t}]A \mathrel{|\!\sim} wlp([\mathbf{s},\mathbf{t}]A, [\mathbf{t}]\varphi) \equiv [\mathbf{t}]\varphi,$$

for arbitrary formulae φ.
Proof: Follows immediately from propositions 5.3, 5.5, and corollary 5.4.□

Proposition 5.5 provide means for an elegant and straightforward definition of successor state axioms in PMON. We can simply apply *wlp* to every action and fluent, and thus get the biconditional. However, this does not solve the problem of regression for nondeterministic actions.

6 Regression

With our definition *wlp* we can now construct regression procedures for pre- and postdiction in PMON. One single procedure for both pre- and postdiction does not exist, and we will argue that *wlp* is applicable for prediction, while wlp^* should be used for postdiction. The way in which observations are handled also differs between the two.

With prediction we mean that, given a scenario, Υ, we want to know if a formula, φ, holds *after* the actions of the scenario are executed. For postdiction we want to know if φ holds at the initial time point of the scenario. For readability, we only consider the cases where no observations occur after the query for prediction and before the query for postdiction.

Algorithm 6.1 (Prediction)
Input is a formula, $[\mathbf{t}]\varphi$, a scenario, Υ, and a time point $\mathbf{s}$, such that no observation, or starting point of an action, in Υ occurs at a time point $< \mathbf{s}$. Output is a formula β such that $\Upsilon \mathrel{|\!\sim} [0]\beta \rightarrow [\mathbf{t}]\varphi$ holds. β describes all initial states such that, if the action sequence is started in either of them, the sequence terminates in a state satisfying φ.

1. $\tau \leftarrow \mathbf{t}$ and $\alpha \leftarrow \varphi$.
2. Repeat the following steps until a formula is returned.
   (a) If there is an observation $[\tau]\delta$ in Υ, then $\alpha \leftarrow (\delta \rightarrow \alpha)$.
   (b) If $\tau = \mathbf{s}$ then return α.
   (c) If there is an schedule statement $[\mathbf{t},\tau]A$ in Υ, then $\alpha \leftarrow wlp([\mathbf{t},\tau]A, [\tau]\alpha)$ and $\tau \leftarrow \mathbf{t}$, else $\tau \leftarrow \tau - 1$ and $\alpha \leftarrow [\tau-1]\alpha[]$.

□

If Υ is deterministic, then $\Upsilon \mathrel{|\!\sim} [0]\beta \equiv [\mathbf{t}]\varphi$ holds for any β returned by the algorithm.
Correctness follows from proposition 5.3.
We illustrate algorithm 6.1 briefly with the scenario in example 3.1 slightly changed: We do not include OBS2. Our goal is to prove that $H(6, white)$ is a consequent of the scenario.
Input: $\varphi \equiv H(6, white)$, $\Upsilon =$ the changed scenario, and $\mathbf{s} = 0$.
First Iteration: $\tau = 6$ and $\alpha \equiv H(6, white)$.
Second Iteration: $\tau = 5$ and $\alpha \equiv H(5, white)$.
Third Iteration: The schedule statement $[3,\tau]Drop$ is found, and $wlp([3,5]Drop, Holds(5, white))$ is computed. After the iteration $\tau = 3$ and $\alpha \equiv \neg H(3, over\_board) \wedge H(3, white)$.
Sixth Iteration $\tau = 0$ and $\alpha \equiv \neg H(0, over\_board) \wedge H(0, white)$.
In the sixth iteration the observation at time point 0 is considered, so

$$\alpha \equiv \\ (H(0, over\_board) \wedge \neg H(0, white) \wedge \neg H(0, black)) \rightarrow \\ (\neg H(0, over\_board) \wedge H(0, white)),$$

which is returned by the algorithm.

The returned formula is a contradiction, which tells us that the goal, H(6,white), is not a consequence of the scenario. If the returned formula is a tautology, the goal is a consequent. The third case, when the formula is neither a contradiction, nor a tautology, implies that the formula is a *condition* at the initial time point for the scenario to entail the goal.

For the postdiction case, we start off by taking the last observation and regress back to the initial time point. We are now interested in regression results that are implied by the conditions *after* the action is performed, since the observations are parts of the axiomatization. We will therefore make use of corollary 5.4 for this algorithm.

Algorithm 6.2 (Postdiction)
Input is a formula, $[0]\varphi$, a consistent action scenario description, Υ, and a time point $\mathbf{s}$, where an observation, $[\mathbf{s}]\delta$, occurs, and no observation occurs at any time point $> \mathbf{s}$. Output is a formula β such that every initial state that is consistent with the scenario, and no other state, is a model of β.

1. $\tau \leftarrow \mathbf{s}$ and $\alpha \leftarrow \mathbf{T}$.
2. Repeat the following steps until a formula is returned.
   (a) If there is an observation $[\tau]\delta$ in Υ, then $\alpha \leftarrow (\delta \wedge \alpha)$.
   (b) If $\tau = \mathbf{0}$ then return α.
   (c) If there is a schedule statement $[\mathbf{t}, \tau]A$ in Υ, then $\alpha \leftarrow wlp^*([\mathbf{t}, \tau]A, [\tau]\alpha)$ and $\tau \leftarrow \mathbf{t}$, else $\tau \leftarrow \tau - 1$ and $\alpha \leftarrow [\tau - 1]\alpha[]$.

□

When a formula β has been generated we can choose if we want to prove that the scenario is consistent with the initial states φ by proving that $\varphi \rightarrow \beta$ holds. If we want to prove that $\Upsilon \mid\!\sim [0]\varphi$, we prove that $\beta \rightarrow \varphi$ holds.
Correctness follows from corollary 5.4.
To illustrate algorithm 6.2, we again look at example 3.1. This time we remove OBS1, and input OBS2, $H(6, white) \wedge \neg H(6, black)$, to the algorithm. Without going into details, we can note that

$$wlp^*([3,5]Drop, H(5, white) \wedge \neg H(5, black)) = \neg H(3, over\_board) \wedge H(3, white) \wedge \neg H(3, black),$$

which coincide with our intuition.

The intuition for the difference of how observations are handled by the algorithms is that for prediction, we want observations to *verify* the computation results, i.e. to described state sets that are subsets of the regression result, and, for postdiction, the observation should filter out all state not being consistent with it.

7 Conclusion and Future Work

We have presented a computational strategy for prediction and postdiction for a large subset of PMON. The strategy is based on Dijkstra's weakest liberal precondition operator. We have shown that *wlp* and its conjugate wlp^* have different properties for nondeterministic scenarios and that the former is applicable for prediction, and the latter for postdiction. For deterministic scenarios, the two operators coincide, thus providing a formal foundation for generating successor state axioms.

The formalism used and the results provided in this paper provide solutions to three problems found in the work by Fangzhen Lin. First, the choice of PMON dealswith the problem of explicit enumeration of effects of nondeterministic action. Secondly, *wlp* enables regression over nondeterministic actions without any oracle features or transformations to deterministic action. Thirdly, *wlp* handles nondeterministic actions with conditional effects properly.

We think that this approach is promising, and we will use *wlp* for implementations of a PMON planner, soon to appear online.

Acknowledgements
Thanks to Patrick Doherty, Thomas Drakengren, Silvia Coradeschi, and Joakim Gustafsson for help and comments.

References

[Dijkstra and Scholten, 1990] W. D. Dijkstra and C. S. Scholten. *Predicate Calculus and Program Semantics.* Springer-Verlag, 1990.

[Doherty, 1994] P. Doherty. Reasoning about action and change using occlusion. In *ECAI'94*, 1994.

[Doherty, 1996] P. Doherty. PMON$^+$: A Fluent Logic for Action and Change – Formal Specification. Tech. Report, 1996. Available at `http://www.ida.liu.se/labs/kplab/`.

[Gustafsson and Doherty, 1996] J. Gustafsson and P. Doherty. Embracing Occlusion in Specifying the Indirect Effects of Actions. In *KR'96*, 1996.

[Lin, 1996] F. Lin. Embracing Causality in Specifying the Indeterminate Effects of Actions. In *AAAI'96*, 1996.

[Lukaszewicz and Madalinska-Bugaj, 1995] W. Lukaszewicz and E. Madalinska-Bugaj. Reasoning about action and change using Dijkstra's semantics for programming languages: Preliminary report. In *IJCAI'95*, 1995.

[Pednault, 1986] E. P. D. Pednault. *Toward a Mathematical Theory of Plan Synthesis.* PhD thesis, Dept. of Electrical Engineering, Stanford University, December 1986.

[Reiter, 1991] R. Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 418–420. Academic Press, San Diego, 1991.

[Sandewall, 1994] E. Sandewall. *Features and Fluents. The Representation of Knowledge about Dynamical Systems, volume I.* Oxford University Press, 1994. ISBN 0-19-853845-6.

[Waldinger, 1977] R. Waldinger. *Achieving Several Goals Simultaneously*, volume 8 of *Machine Intelligence*, pages 94–136. Ellis Horwood, 1977.