Extending dynamic logic with refinements of abstract actions

NICO ROOS, *Maastricht University, Data Science and Knowledge Engineering, P.O. Box 616, 6200 MD Maastricht, The Netherlands.*
E-mail: roos@maastrichtuniversity.nl

Abstract

Humans can make a plan by refining abstract actions. Dynamic logic, which enables reasoning about the dynamics of actions, does not support this form of planning. This paper investigates the extension of dynamic logic with a refinement relation that specifies how an abstract action can be refined into a more specific (composite) action. The paper investigates the properties of the refinement relation, the derivation of the refinement relation and a proof system based on a prefixed-tableau.

1 Introduction

Dynamic logic [11, 30] is a formal logic that enables reasoning about the dynamics of actions. Dynamic logic can be used for planning by describing the effects of action executions (the effect axioms) and complementing this with a mechanism for handling the frame problem, such as the addition of frame axioms [8, 29, 39] or a preference relation over models [39]. Dynamic logic allows for more expressiveness than the STRIPS [10] based planners. The additional expressiveness comes with a price, a higher worst case computational complexity. Another difference between STRIPS and dynamic logic is that the former is a planning system while the latter is a formal logic for reasoning about the dynamics of actions.

A form of planning that is not supported by dynamic logic, is planning through the refinement of abstract actions. An abstract action is an action that cannot be executed directly. It needs to be replaced by a more specific plan whose execution realizes the abstract action. For instance, 'going to Rome' is an abstract action that requires a more specific (refined) plan of ' taking the bus to the train station' followed by 'taking the train to Rome' to realize it.

The philosopher Bratman argued that humans plan through the refinement of abstract actions [3]. His beliefs, desires and intentions (BDI) model implements his ideas about planning through the refinement of abstract actions. The BDI model assumes that an agent has several, possibly mutually inconsistent, desires from which the agent chooses, after some deliberation, one desire that the agent will intend to realize (the intention). The desires and the chosen intention are abstract actions in Bratman's BDI model. Next, the agent considers several refinement options of the abstract intention and chooses one to them to become a refined intention. Since intentions and desires are (abstract) actions, Bratman's BDI model is essentially planning through the refinement of abstract actions.

Several logics that try to formally describe the BDI model have been proposed. For an overview, see for instance [17, 26]. Most noticeable are the logics proposed by Cohen and Levesque [5] and by Rao and Georgeff [33]. However, these works are not considering the refinement of abstract actions. Both Cohen and Levesque [5] and Rao and Georgeff [33] focus on formalizing the meaning of an intention.

A hierarchical task network (HTN) [13, 36] is as a practical planning approach that implements planning through the refinement of abstract actions. HTNs have been used as a practical way to implement Bratman's BDI model [37]. An HTN consists of a set of tasks divided into primitive

Vol. 32, No. 7, © The Author(s) 2022. Published by Oxford University Press.

Advance Access Publication on 21 February 2022 https://doi.org/10.1093/logcom/exac003

tasks and abstract tasks. A primitive task can be executed by a corresponding action provided that the precondition of the action is met, and an abstract task can be refined by some method provided that the precondition of the method is met. The method specifies a (partially) ordered sequence of less abstract tasks.

An HTN is a planning system but not a formal logic like dynamic logic. Not being a formal logic hampers the analysis of HTNs. Herzig *et al.* [18] extend propositional dynamic logic (PDL) with a refinement relation in order to provide a semantics for HTNs. They translate an HTN into the extension of PDL and formulate rationality postulates that guarantee the soundness and completeness of refinements of abstract tasks. No reasoning process is provided. The authors claim that their extension of PDL is undecidable because of correspondences with undecidable grammar logics, but no proof is given for the claim. The possible undecidability of the extension of PDL is not a problem for the rationality postulates because the postulates rely on standard PDL.

Zhou and Zhang [40] were the first to propose an extension of PDL with abstract actions and refinements. They define a subsumption relation over actions, and an operator denoting the abstract action of achieving that a proposition holds. They investigate several properties of these added operators. However, no proof system is given for the extended PDL. Zhou and Zhang [40] consider a proof system for the refinement relation to be very difficult.

This paper is based on and extends Section 4 of a paper by Roos [35]. In the latter paper, also a subsumption relation over actions is introduced. There is an important difference in terminology between Roos' paper [35] and Zhou and Zhang's paper [40]. Zhou and Zhang say that a specific action subsumes an abstract action because the effects of a specific action are a superset of the effects of an abstract action. Roos used the terminology from description logics [19], where the relation describing an abstract action subsumes the relation describing a specific action because the former is a superset of the latter. In order to avoid confusion, this paper will use the term *refinement relation*.

This paper presents an extension of PDL with a refinement relation over actions (PDLR). The extension is similar to the one proposed by Zhou and Zhang [40], Roos [35] and Herzig *et al.* [18]. To ensure that the refinement relation specifies a refinement of an abstract action into a composite action (plan) consisting of less abstract atomic actions, we only allow for the specification of refinements of atomic actions. If we could specify that one composite action refines another composite action, then this can semantically be interpreted as specifying alternative plans. For instance, the plan $a; b$ refines the plan $c; d$ where a, b, c and d are atomic actions. An alternative plan does not describe a refinement of an abstract action. The restriction on the refinement relation also enables us to define which actions are abstract actions. An action a is an abstract action if it can be refined, unless each refinement is an atomic action b and a is a refinement of b. The last condition is needed because identical actions refine each other.

This paper also investigates the derivation of the refinement relation from other information. This investigation leads to the introduction of a $\overline{\ }$-operator that modifies a modal necessity operator $\mathcal{N}$. Semantically $\overline{\mathcal{N}}\varphi$ will be used to denote that the proposition φ characterizes all the states that can be reached using the relation $R^{\mathcal{N}}$ associated with the modal operator $\mathcal{N}$. We can read $\overline{\mathcal{N}}$ as '*only* $\mathcal{N}$'. So, given the modal operator K of epistemic logic, $\overline{K}\varphi$ is read as 'only knowing φ'. Finally, this paper presents a proof system based on the construction of a prefixed tableau and proves the correctness and completeness of the proof system.

The remainder of the paper is organized as follows. The next, rather long, section contains the preliminaries. It introduces PDL and discusses the main aspects of a proof system based on the construction of a prefixed tableau. Section 3 introduces abstract actions and a refinement relation over actions. Section 4 addresses the derivation of the refinement relation over actions using the

effects of the actions. Section 5 presents a proof system for the proposed extension of dynamic logic. Section 6 discusses the relations with agent programming languages and with HTNs, and Section 7 concludes the paper.

2 Preliminaries

PDL This paper focuses on PDL [11]. PDL has a language $\mathcal{L}$ that is defined starting from a set of atomic actions A and a set of atomic propositions $\mathcal{P}$.

Composite actions (plans) are formulated using regular expressions that also include test-actions and a 'no-operation' action. The following operators are used to construct composite actions:

- an is-followed-by operator: ; ,
- a non-deterministic choice operator: + ,
- an iteration operator: * and
- a test operator: ? .

The composite action $\alpha;\beta$ denotes that β is executed after α, the composite action $\alpha+\beta$ denotes that a non-deterministic choice is made between executing α and β, the composite action α^* denotes that α is executed 0 or more times, i.e.$\alpha^* = \varepsilon + \alpha + (\alpha;\alpha) + (\alpha;\alpha;\alpha) + (\alpha;\alpha;\alpha;\alpha) + \ldots$ where ε represents *no-operation*, and the composite action $\varphi?$ denotes a test that succeeds if the proposition φ holds. We will use the Greek characters $\alpha, \beta, \ldots$ for composite actions, and the roman characters $a, b, \ldots$ for atomic actions. Of course, composite actions also include the atomic actions.

The propositions are defined using a finite set of atomic propositions $\mathcal{P}$, the special propositions $\top$ and $\bot$ representing *true* and *false*, respectively, the logical operators $\neg$, $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$, and the modal operators $[\alpha]$, $\langle\alpha\rangle$, $\Box$ and $\Diamond$. The modal operators will be explained below.

The language $\mathcal{L}$ of propositions φ and the set $\mathcal{A}$ of composite actions α are defined by a double recursion. The double recursion is needed because the tests in $\mathcal{A}$ depend on the language $\mathcal{L}$ and the modal propositions $[\alpha]\varphi$ and $\langle\alpha\rangle\varphi$ depend on the composite actions $\mathcal{A}$.

$$\varphi ::= \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \varphi \leftrightarrow \varphi \mid [\alpha]\varphi \mid \langle\alpha\rangle\varphi \mid \Box\varphi \mid \Diamond\varphi \text{ with } p \in \mathcal{P}$$
$$\alpha ::= \varepsilon \mid a \mid \alpha;\alpha \mid \alpha + \alpha \mid \alpha^* \mid \varphi? \text{ with } a \in A.$$

The semantics of PDL is defined using interpretations that are Kripke structures. An interpretation $\mathcal{I} = (S, \pi, R^A)$ contains a set of states S describing the possible states of the environment, an interpretation function $\pi : S \rightarrow (\mathcal{P} \rightarrow \{\mathsf{true}, \mathsf{false}\})$ specifying for each state the truth-value of every atomic proposition, and a function $R^A : A \rightarrow 2^{S\times S}$ specifying an accessibility relation for each action in A. The accessibility relation $R^A(a)$ denotes that one of the states $\{t \mid (s,t) \in R^A(a)\}$ will be reached when executing action $a \in A$ in the state $s \in S$.

The interpretation of the composite actions in $\mathcal{A}$ is specified by the function $R^{\mathcal{A}} : \mathcal{A} \rightarrow 2^{S\times S}$, which specifies an accessibility relation $R^{\mathcal{A}}(\alpha) \subseteq S \times S$ for every composite action $\alpha \in \mathcal{A}$.

- $R^{\mathcal{A}}(a) = R^A(a)$ for every $a \in A$
- $R^{\mathcal{A}}(\alpha;\beta) = R^{\mathcal{A}}(\alpha) \circ R^{\mathcal{A}}(\beta) = \{(s,s'') \mid (s,s') \in R^{\mathcal{A}}(\alpha), (s',s'') \in R^{\mathcal{A}}(\beta)\}$
- $R^{\mathcal{A}}(\alpha+\beta) = R^{\mathcal{A}}(\alpha) \cup R^{\mathcal{A}}(\beta)$
- $R^{\mathcal{A}}(\varepsilon) = \{(s,s) \mid s \in S\}$
- $R^{\mathcal{A}}(\alpha^*) = \bigcup_{k=0}^{\infty} R^{\mathcal{A}}(\alpha^k)$ where $R^{\mathcal{A}}(\alpha^k) = R^{\mathcal{A}}(\alpha^{k-1}) \circ R^{\mathcal{A}}(\alpha)$ and $R^{\mathcal{A}}(\alpha^0) = R^{\mathcal{A}}(\varepsilon)$
- $R^{\mathcal{A}}(\varphi?) = \{(s,s) \mid (\mathcal{I},s) \models \varphi\}$

The possible and necessary effect of executing an action α described by the proposition φ, can be specified by the propositions $[\alpha]\varphi$ and $\langle a\rangle\varphi$, respectively. The semantics of these propositions is defined w.r.t. to a state $s \in S$ of an interpretation $\mathcal{I}$:

- $(\mathcal{I}, s) \models [\alpha]\varphi$ iff for every $s' \in S$, if $(s, s') \in R^{\mathcal{A}}(\alpha)$, then $(\mathcal{I}, s') \models \varphi$.
- $(\mathcal{I}, s) \models \langle\alpha\rangle\varphi$ iff for some $s' \in S$, $(s, s') \in R^{\mathcal{A}}(\alpha)$ and $(\mathcal{I}, s') \models \varphi$.

We will use the propositions $\Box\varphi$ and $\Diamond\varphi$ containing the modal operators $\Box$ and $\Diamond$, to specify that the proposition φ *always* and *sometimes* holds, respectively. For instance, $\Box(\varphi \rightarrow (\langle\alpha\rangle\psi \wedge [\alpha]\psi))$ specifies that in every situation where φ holds, ψ is the possible and necessary effect of executing α. Semantically, we do not model $\Box$ by considering all states in S. Instead, we will interpret $\Box$ as referring to all states reachable from the current state by some sequence of actions.[1] This makes it possible to express uncertainty about what always holds and makes it possible to specify a condition under which a proposition always holds. The semantics of $\Box$ denoting *always* and its dual $\Diamond$ denoting *sometimes* w.r.t. to a state $s \in S$ of an interpretation $\mathcal{I}$ is defined by

- $(\mathcal{I}, s) \models \Box\varphi$ iff for every $s' \in S$, if $(s, s') \in \left(\bigcup_{a\in A} R^A(a)\right)^*$, then $(\mathcal{I}, s') \models \varphi$;
- $(\mathcal{I}, s) \models \Diamond\varphi$ iff for some $s' \in S$, $(s, s') \in \left(\bigcup_{a\in A} R^A(a)\right)^*$ and $(\mathcal{I}, s') \models \varphi$.

We will use $\Sigma \subseteq \mathcal{L}$ to describe all available information. We assume that Σ is a finite set of propositions. The entailment relation between the information $\Sigma \subseteq \mathcal{L}$ and a conclusion $\varphi \in \mathcal{L}$ is given by

- $\Sigma \models \varphi$ iff for every interpretation $\mathcal{I}$ and for every state $s \in S$ of that interpretation, $(\mathcal{I}, s) \models \Sigma$ implies $(\mathcal{I}, s) \models \varphi$. Here, $(\mathcal{I}, s) \models \Sigma$ denotes $(\mathcal{I}, s) \models \sigma$ for every $\sigma \in \Sigma$.
- $\models \varphi$ is a shortcut for $\varnothing \models \varphi$.

A proof system for PDL One possible proof system for PDL is based on the construction of a prefixed tableau described by De Giacomo and Massacci [14, 15]. It is not the tableau method with the best worst-case time complexity, but it has the advantage that it can be combined with prefixed tableaux for other modal operators [24] and that it is a more traditional tableau method that can be implemented using from left to right depth-first search [20]. Tableau methods for PDL with an optimal worst-case time complexity are described in [16, 28].

In a prefixed tableau, a prefix is associated with every proposition φ in the semantic tableau. For instance, $0.\langle a\rangle.3.\langle b\rangle.9.\langle c\rangle.11 : \varphi$. The prefix $0.\langle a\rangle.3.\langle b\rangle.9.\langle c\rangle.11$ can be viewed as a sequence of states together with a specification of the accessibility relation between the states. The numbers in the prefix are id-numbers of states generated by the semantic tableau rules: s_0, s_3, s_9 and s_{11}. Moreover, the symbols a, b and c between '$\langle$' and '$\rangle$' are atomic actions representing the instances (s_0, s_3), (s_3, s_9) and (s_9, s_{11}) of the accessibility relations $R^A(a)$, $R^A(b)$ and $R^A(c)$, respectively. In the last state s_{11} of the prefix, the proposition φ holds.

We will use s^x to denote the last state s_n specified by a prefix $x = y.\langle a\rangle.n$ with $n \in \mathbb{N}$, $a \in A$ and y being a prefix. This avoids the need to talk about the structure of the prefix x. We will use $x = y.z$ to denote that the prefix x is the result of extending the prefix y with several additional state transitions and states, denoted by z.

A semantic tableau is a tree in which nodes represent sets of prefixed propositions. For convenience, we will not make an explicit distinction between a node and the set of prefix-

[1] In (2), $\Box$ is called a *global modality* if it refers to all states in S and $\Box$ is called a *master modality* if it refers to all states reachable from the current state. Here, $\Box$ is defined as a master modality, and its use is similar to the use of $\Box$ in the modal situation calculus $\mathcal{ES}$ (21).

propositions it represents. This enables us to talk about a node Γ while Γ is also the set of prefixed propositions represented by the node. We will use the notion of a *saturated* node in the construction of a semantic tableau. A node is saturated if the node cannot be extended by the application of a rule. It is therefore always a leaf of the tableau.

The following rules can be used to construct a semantic tableau for dynamic logic. A rule may only be applied if the antecedent is present in the leaf that we currently consider and if the consequence is not yet present in the leaf.

$$\frac{x{:}[a]\varphi}{x.\langle a\rangle.n{:}\varphi} \qquad \frac{x{:}\langle a\rangle\varphi}{x.\langle a\rangle.n{:}\varphi} \qquad \frac{x{:}[\psi?]\varphi}{x{:}\neg\psi \mid x{:}\psi, x{:}\varphi} \qquad \frac{x{:}\langle\psi?\rangle\varphi}{x{:}\psi, x{:}\varphi}$$

$$\frac{x{:}\neg[\alpha]\varphi}{x{:}\langle\alpha\rangle\neg\varphi} \qquad \frac{x{:}\neg\langle\alpha\rangle\varphi}{x{:}[\alpha]\neg\varphi} \qquad \frac{x{:}[\alpha;\beta]\varphi}{x{:}[\alpha][\beta]\varphi} \qquad \frac{x{:}\langle\alpha;\beta\rangle\varphi}{x{:}\langle a\rangle\langle\beta\rangle\varphi}$$

$$\frac{x{:}[\alpha+\beta]\varphi}{x{:}[\alpha]\varphi, x{:}[\beta]\varphi} \qquad \frac{x{:}\langle\alpha+\beta\rangle\varphi}{x{:}\langle\alpha\rangle\varphi \mid x{:}\langle\beta\rangle\varphi} \qquad \frac{x{:}[\alpha^*]\varphi}{x{:}\varphi, x{:}[\alpha][\alpha^*]\varphi} \qquad \frac{x{:}\langle\alpha^*\rangle\varphi}{x{:}\varphi \mid x{:}\langle\alpha\rangle\langle\alpha^*\rangle\varphi}$$

There are additional conditions on the application of the first two rules on the first row.

- The rule $\frac{x{:}[a]\varphi}{x.\langle a\rangle.n{:}\varphi}$ may only be applied if $a \in A$ is an atomic action, and the prefix $x.\langle a\rangle.n$ is already present in the current leaf.
- The rule $\frac{x{:}\langle a\rangle\varphi}{x.\langle a\rangle.n{:}\varphi}$ may only be applied if $a \in A$ is an atomic action, and the prefix $x.\langle a\rangle.n$ does *not* exist in the current leaf. We introduce a new state s_n represented by $x.\langle a\rangle.n$ where $n \in \mathbb{N}$ is a new number. There is another condition on the application of $\frac{x{:}\langle a\rangle\varphi}{x.\langle a\rangle.n{:}\varphi}$, which will be explained below.

It is not difficult to see that a proposition $x : \langle a^*\rangle\varphi$ together with the rules $\frac{x{:}\langle a^*\rangle\varphi}{x{:}\varphi \mid x{:}\langle a\rangle\langle a^*\rangle\varphi}$ and $\frac{x{:}\langle a\rangle\varphi}{x.\langle a\rangle.n{:}\varphi}$ will result in an infinitely long branch in a semantic tableau because for every possible number of iterations, a new prefix will be created. Infinitely long branches can also arise through other interactions between rules for iterations and rules introducing new states. We can avoid the infinitely long branches using a *blocking mechanism*. Specifically, we can block the creation of longer prefixes $y = x.z$ if the set of propositions with prefix x: $\Gamma(x) = \{\varphi \mid x : \varphi \in \Gamma\}$ is the same as the set of propositions with prefix y: $\Gamma(y) = \Gamma(x)$. The longer prefix y provides no additional information, i.e. $\Gamma(x)$ is satisfiable if and only if $\Gamma(y)$ is satisfiable. If $y = x.z$ and $\Gamma(y) = \Gamma(x)$, we say that y is blocked by x.

To see why the blocking mechanism, avoids infinitely long branches in the semantic tableau, note that the first node of a branch contains a finite set of propositions Σ with prefix 0. If we ignore the prefixes, only a finite set of propositions Ω can be derived by applying the tableau rules.[2] Therefore, for every node Γ of the branch and for every prefix x, the set of propositions with prefix x in Γ will be a finite subset of Ω, i.e. $\Gamma(x) = \{\varphi \mid x : \varphi \in \Gamma\} \subset \Omega$. Let the number of actions mentioned in a prefix x denote the length $||x||$ of the prefix. Since $\Gamma(x)$ is finite, only a finite number of new prefixes of length $||x|| + 1$ can be derived form $\Gamma(x)$ using the tableau rule $\frac{x{:}\langle a\rangle\varphi}{x.\langle a\rangle.n{:}\varphi}$. Hence, an infinitely long branch is only possible if there is no restriction on the length of a prefix. Since there are $2^{|\Omega|}$ different subsets of Ω, a prefix y with $||y|| > 2^{|\Omega|}$ must have proper sub-prefix x such that $y = x.z$ and $\Gamma(x) = \Gamma(y)$. Hence, the blocking mechanism ensures that all branches will have a finite length.

The blocking mechanism raises an issue concerning the satisfiability of the propositions $\Gamma(y)$ if y is blocked by x. Clearly, the propositions in $\Gamma(y)$ are satisfiable iff the propositions in $\Gamma(x)$ are satisfiable. To address the satisfiability of $\Gamma(y)$, we must distinguish two cases:

[2]Note that Ω is related to but different from the Fischer–Ladner (11) closure.

1. the blocked prefix represents an infinitely long path of states, and
2. the blocked prefix represents an infinite number of execution paths of finite lengths: $\alpha^* = \varepsilon + \alpha + (\alpha;\alpha) + (\alpha;\alpha;\alpha) + (\alpha;\alpha;\alpha;\alpha) + \ldots$.

In the former case, if the leaf is not closed, then there must be an interpretation containing an infinitely long path of states. Such a path can be represented by a cycle in the accessibility relation.

In the latter case, $\Gamma(x)$ contains the proposition $\langle\alpha^*\rangle\varphi$, and the prefix y is the result of executing the (composite) action α at least one more time. Note that the tableau rule $\frac{x:\langle\alpha^*\rangle\varphi}{x:\varphi|x:\langle\alpha\rangle\langle\alpha^*\rangle\varphi}$ ensures that every finite length execution path results in a different branch. We are interested in the shortest execution path $(\alpha;\alpha;\cdots;\alpha)$ after which φ holds. Hence, a branch containing $\Gamma(y)$ with $y = x.z$ and $\Gamma(y) = \Gamma(x)$ cannot describe shortest execution path and can therefore be ignored. So, the satisfiability of $\Gamma(x)$ is determined by an alternative branch containing $x : \varphi$.

To determine whether we have the second case mentioned above if x is blocking y, we must identify whether the prefix y is generated by $x : \langle\alpha^*\rangle\varphi$. We cannot simply check whether $\langle\alpha^*\rangle\varphi \in \Gamma(y) = \Gamma(x)$. The proposition $\langle\alpha^*\rangle\varphi \in \Gamma(y)$ need not follow from $\langle\alpha^*\rangle\varphi \in \Gamma(x)$. To give an illustration, consider the tableau generated by the set of propositions: $\{\neg p, [b^*]\langle a^*\rangle p, [b^*]\langle b\rangle\neg p\}$ [15]. De Giacomo and Massacci [14, 15] propose to introduce names for propositions of the form $\langle\alpha^*\rangle\varphi$ to distinguish the different occurrences. They replace the last rule on the last row by two rules.

$$\frac{x:\langle\alpha^*\rangle\varphi}{x:\chi,}, M(\chi) = \langle\alpha^*\rangle\varphi \qquad \frac{x:\chi}{x:\varphi|x:\langle\alpha\rangle\chi}$$

The left rule introduces a *new* atomic proposition $\chi \notin \mathcal{L}$. The rule may only be applied if no new proposition χ has been introduced for $x : \langle\alpha^*\rangle\varphi$ yet. However, if the prefixes are different but the proposition is the same, e.g. $x : \langle\alpha^*\rangle\varphi$ and $y : \langle\alpha^*\rangle\varphi$, then a new proposition *must* be introduced for both $x : \langle\alpha^*\rangle\varphi$ and $y : \langle\alpha^*\rangle\varphi$, i.e. $x : \chi$ and $y : \chi'$. The new atomic proposition χ can be seen as a name of $\langle\alpha^*\rangle\varphi$. We use a mapping $M(\chi) = \langle\alpha^*\rangle\varphi$ to register the introduced name χ for $\langle\alpha^*\rangle\varphi$, and we use $\mathcal{X}$ to denote the set of new names that have been introduced.

Given the two new rules, we can give the definition of the blocking mechanism.

- Let x be a prefix.
  $\overline{\Gamma}(x) = \{\overline{\varphi} \mid \varphi \in \Gamma(x)\}$ and $\overline{\varphi}$ is the result of replacing every occurrence of $\chi \in \mathcal{X}$ in φ by $M(\chi)$.
- A prefix y is blocked by a prefix x in Γ iff:
  - x is an initial part of y, i.e. $y = x.z$, and
  - the prefix x is already blocked, *or* $\overline{\Gamma}(x) = \overline{\Gamma}(y)$.
- A prefix y is *directly* blocked by a prefix x iff y is blocked by x and no proper sub-prefix of y is blocked.

If a prefix is blocked, we may not apply the second rule on the first row of the above. The tableau rule:

$$\frac{x : \langle a\rangle\varphi}{x.\langle a\rangle.n : \varphi}$$

may only be applied if the above listed conditions are met and if the prefix x is *not blocked*.

The introduction of names enables us to decide whether we have the first or the second case of blocking described above. In the second case, as argued above, we can *ignore* the leaf of a branch.

A leaf Λ of the tableau is *ignorable* iff

- Λ is saturated,
- Λ is not closed, and
- there is a prefix x that directly blocks a prefix y, and for some new proposition $\chi \in \mathcal{X}$, $\{y : \chi, x : \chi\} \subseteq \Lambda$.

The introduction of ignorable leafs also requires an adaptation of the definition of a closed semantic tableau.

A semantic tableau is closed iff every branch is closed or can be ignored.

Using this definition of a closed tableau, we can use the semantic tableau for a standard refutation proof. De Giacomo and Massacci [14, 15] proved the correctness and completeness of the semantic tableau method.

Tableau rules for always and sometimes The modal operators for always ($\Box$) and sometimes ($\Diamond$) do not belong to standard PDL and De Giacomo and Massacci do not define tableau rules for these operators. In Subsection 5.6, the tableau rules for these operators will be defined.

3 Refinement relations over actions[3]

Semantically, an action a in dynamic logic describes a transition from a state s to one or more others states s'. A transition to multiple states indicates uncertainty about the effect of executing action a in the state s. A refinement of the action a is a (composite) action that selects a subset of the transitions associated with the action a [18, 35, 40]. The refinement provides a more detailed specification of executing a. We say that the atomic action a is an *abstract action* if it can be refined, unless each refinement is an atomic action b and a is a refinement of b. The last condition is needed because identical actions refine each other.

Since there can be multiple refinements of an abstract action, even if there is no uncertainty in the transition of each refinement, the abstract action covering them must have multiple transitions. Because an abstract action can only be executed through the execution of a refinement, the uncertainty expressed by the multiple transitions of the abstract action, expresses uncertainty about which refinement is chosen. If there is uncertainty about the effect of executing a refinement (about the state that is reached by executing a refinement), then the multiple transitions of an abstract action capture both the uncertainty in the execution of a refinement and the uncertainty about which refinement is chosen.

Figure 1 gives an illustration of abstract actions and refinements. In the example, there is no uncertainty about the effects of executing a composite action that cannot be refined any further. Figure 1 shows six atomic actions: ‘walk’, ‘take the bus’, ‘take the train’, ‘go to Rome by train’, ‘go to Rome by car’ and ‘go to Rome’. The action ‘go to Rome’ is the most abstract action, which is refined into ‘go to Rome by train’ and ‘go to Rome by car’. The former is further refined into the composite action ‘walk; take the train’ and ‘take the bus; take the train’. Although we can present the five atomic actions in dynamic logic, we cannot specify the refinement relation over these actions.

[3] This section is based on Subsection 4.1 in [35] and overlaps with [40].

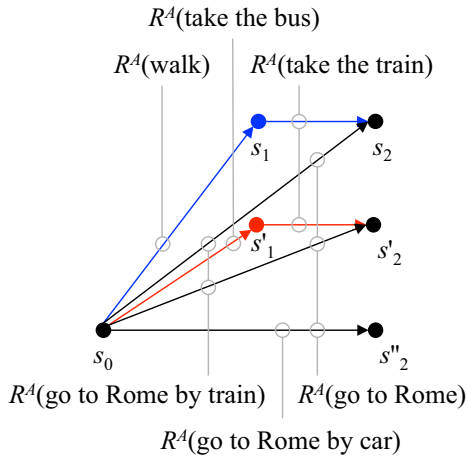


FIGURE 1 The semantics of behaviour.

Here, we propose a refinement relation over actions. We will use the operator $\sqsubseteq$ to denote this relation.[4] So, $\alpha \sqsubseteq b$ denotes that the composite action α refines the (abstract) atomic action b. We will denote PDL extended with a refinement relation over actions by PDLR.

The idea behind defining a refinement relation over actions is the following: if the states that can be reached by executing an action α in a state s, is a subset of the states that can be reached by executing an action β in the state s, then the action α refines the action β in the state s. We denote that α refines β by $\alpha \sqsubseteq \beta$. The semantics of the refinement relation $\sqsubseteq$ over actions given a state s, is specified by the following definition.

DEFINITION 3.1
Let $\alpha \in \mathcal{A}$ and $\beta \in \mathcal{A}$ be two actions.
$(\mathcal{I}, s) \models \alpha \sqsubseteq \beta$ iff for every $t \in S$, if $(s,t) \in R^{\mathcal{A}}(\alpha)$, then $(s,t) \in R^{\mathcal{A}}(\beta)$.

The refinement relation $\alpha \sqsubseteq \beta$ does not imply that the composite action β is an abstraction of α. The composite action β might be an alternative plan that results in transitions to the same states as the action α. To give an illustration, suppose that $s_2 = s'_2$ in the example in Figure 1. Then 'walk; take the train' is an alternative plan for 'take the bus; take the train'. Of course, $s_2 = s'_2$ implies that we cannot encode information about the executed actions in a state.

We are interested in hierarchical refinements of abstract actions. That is, in refinement relations of the form $\alpha \sqsubseteq b$ where $\alpha \in \mathcal{A}$ is a composite action refining the abstract action $b \in A$. We therefore place a restriction on the use of the refinement relation $\sqsubseteq$. We will only allow for refinement relations (as part of larger propositions) of the form $\alpha \sqsubseteq b$ where $\alpha \in \mathcal{A}$ is a (composite) action and $b \in A$ is an atomic action, in the available information $\Sigma \subseteq \mathcal{L}$. It is still possible to derive the more general form $\alpha \sqsubseteq \beta$ from the initial information, i.e. $\Sigma \models \alpha \sqsubseteq \beta$. If $\alpha \sqsubseteq \beta$ is a conclusion derivable from

[4] We use the same notation as is used for denoting refinement of roles (role hierarchies) in description logics [19]. Although roles in description logics are usually described by verbs, these verbs do not denote actions but state properties between pairs of objects.

Σ, then the restriction on refinement relations specified in Σ implies that α must be a hierarchical refinement of β.

Herzig *et al.* [18] claim that PDL extended with a refinement relation is undecidable. They base their claim on the correspondence with grammar logics [7, 9]. However, no proof is given and it is unclear how undecidability results for different grammar logics imply the undecidability of PDL extended with a refinement relation. Although the claim might be correct, the restriction of only allowing a single (abstract) action on right-hand side of $\sqsubseteq$ in the initial information Σ guarantees that the here presented logic is decidable.

The semantic definition of the refinement relation does not imply that the refinement relation between two actions holds in every state of an interpretation $\mathcal{I}$. A refinement relation that depends on the state offers two advantages. Firstly, it enables us to express conditional refinements, and secondly, it enables us to express uncertainty about a refinement relation. Moreover, when combining dynamic logic with a refinement of abstract actions and, for instance, doxastic logic, we can express that one agent believes a refinement while another agent does not.

In order to reason about refinement relations, it would be useful if we could describe a refinement relation in terms of dynamic logic. It turns out that this is possible if we introduce a special set of propositions that are witnesses for states in S.

> Let Ξ be a special set of atomic propositions that is disjoint from the set atomic propositions $\mathcal{P}$ and therefore disjoint from $\mathcal{L}$. The atomic propositions Ξ function as *witnesses* for states in S. There is a bijective function $f : S \to \Xi$ that associates a proposition $\xi \in \Xi$ with every state in S: $f(s) = \xi$, and vice versa: $f^{-1}(\xi) = s$.

Note that Ξ need not be a finite set because S need not be a finite set of states.

Also, note that witnesses differ from nominals in hybrid logic [1]. Nominals are names for states that belong to the language $\mathcal{L}$ and can be used to form new propositions, e.g. $\xi \wedge p$. This is not possible with the witnesses introduced here. We only use a witness ξ as an auxiliary proposition that is used to denote a state that might be reached by executing a (composite) action: $\langle\alpha\rangle\xi$.

PROPOSITION 3.2
Let $\alpha \in \mathcal{A}$ and $\beta \in \mathcal{A}$ be two actions, let $\mathcal{I}$ be an interpretation and let s be a state of the interpretation.

$(\mathcal{I}, s) \models \alpha \sqsubseteq \beta$ iff for every proposition $\xi \in \Xi$, $(\mathcal{I}, s) \models \langle\alpha\rangle\xi$ implies $(\mathcal{I}, s) \models \langle\beta\rangle\xi$.

PROOF. $(\mathcal{I}, s) \models \alpha \sqsubseteq b$, iff for every $t \in S$, if $(s, t) \in R^{\mathcal{A}}(\alpha)$, then $(s, t) \in R^{\mathcal{A}}(\beta)$, iff for every $t \in S$, if $(s, t) \in R^{\mathcal{A}}(\alpha)$ and $(\mathcal{I}, t) \models \xi$, then $(s, t) \in R^{\mathcal{A}}(\beta)$ and $(\mathcal{I}, t) \models \xi$, iff for every ξ, $(\mathcal{I}, s) \models \langle\alpha\rangle\xi$ implies $(\mathcal{I}, s) \models \langle\beta\rangle\xi$. □

The above result will be useful in defining a proof system. By introducing a witness ξ for a state, we will be able to derive, for instance, that

$$(\text{'walk}'; \text{'take the train}') \sqsubseteq \text{'goto Rome}'$$

is implied by the information:

$$\text{'goto Rome by train}' \sqsubseteq \text{'goto Rome}'$$
$$(\text{'walk}'; \text{'take the train}') \sqsubseteq \text{'goto Rome by train}'$$

An (uncountable) infinite set Ξ is not a problem for the proof system. The only place where Ξ plays a role is in the tableau rule for a proposition of the form: $\neg(\alpha \sqsubseteq \beta)$; see Subsection 5.2. This tableau rule introduces a new atomic proposition representing an proposition in Ξ. Since we assume

a finite set of information from which conclusions are derived, only a finite number of elements from Ξ will be considered.

We can also prove the following properties of $\alpha \sqsubseteq \beta$.

PROPOSITION 3.3
Let $(\mathcal{I}, s)$ be an interpretation state pair.

- If $(\mathcal{I}, s) \models \alpha \sqsubseteq \beta$, then for every $\varphi \in \mathcal{L}$, if $(\mathcal{I}, s) \models \langle\alpha\rangle\varphi$, then $(\mathcal{I}, s) \models \langle\beta\rangle\varphi$.
- If $(\mathcal{I}, s) \models \alpha \sqsubseteq \beta$, then for every $\varphi \in \mathcal{L}$, if $(\mathcal{I}, s) \models [\beta]\varphi$, then $(\mathcal{I}, s) \models [\alpha]\varphi$.

PROOF. Since the set of states reachable from s by executing α is a subset of the set of states reachable from s by executing b, the two items of the proposition hold. □

EXAMPLE 3.4
It is not difficult to verify that the following refinement relations hold between the actions in Figure 1:

$(\mathcal{I}, s_0) \models$ 'goto Rome by train' $\sqsubseteq$ 'goto Rome'
$(\mathcal{I}, s_0) \models$ ('walk'; 'take the train') $\sqsubseteq$ 'goto Rome by train'
$(\mathcal{I}, s_0) \models$ ('walk'; 'take the train') $\sqsubseteq$ 'goto Rome'
$(\mathcal{I}, s_0) \models$ ('take the bus'; 'take the train') $\sqsubseteq$ 'goto Rome by train'
$(\mathcal{I}, s_0) \models$ ('take the bus'; 'take the train') $\sqsubseteq$ 'goto Rome'

4 Deriving refinement relations[5]

Proposition 3.3 enables us to apply the refinement relation to derive what holds after executing an action. The converse is not possible. The proposition does not allow us to derive a refinement relation based on the propositions that hold after executing an action.

Characterizing actions by their effects In order to derive a refinement relation, we make the *assumption* that *actions are completely characterized by their effects*. This assumption implies that two actions that have exactly the same effects must be identical. We generalize this to abstract actions and refinements. According to this generalization, an action α refines an action b if the action α realizes at least the same effects as the action b. So, the transitions described by α must be a subset of the transitions described by b. To give an illustration, the action of 'going to Rome by train' refines the action of 'going to Rome' because both actions have as effect 'being in Rome' while the former action also has the effect of 'arriving by train'. The transitions of the former action are a subset of the transitions of the latter action in Figure 1. Note that two atomic actions a and b are identical (describe the same transitions) if a refines b and vice versa.

We can formalize the assumption that actions are completely characterized by their effects as follows.

DEFINITION 4.1
We say that *actions are completely characterized by their effects* in a state s of an interpretation $\mathcal{I}$ iff the following condition holds:

$(\mathcal{I}, s) \models \alpha \sqsubseteq b$ iff for every $\varphi \in \mathcal{L}$, $(\mathcal{I}, s) \models [b]\varphi$ implies $(\mathcal{I}, s) \models [\alpha]\varphi$.

[5]This section is based on Subsection 4.2 in [35].

The semantics of dynamic logic allows for duplicate states. This enables us to construct an interpretation in which two actions a and b have exactly the same effects if executed in a state s, but the transition associated with the two actions are different. If actions are completely characterized by their effects, then Definition 4.1 introduces a restriction on the allowed interpretations, which makes it impossible that a and b are described by different transitions.

The restriction that the assumption that *actions are completely characterized by their effects* places on the allowed interpretations raises two questions.

- Can we always make this assumption? In other words, if an action has at least the same effects as another action, does there exist an interpretation where the former is a refinement of the latter?
- Do propositions that hold after an action execution, but that are not caused by the action, have implications on the applicability of the assumption?

The following proposition addresses the first question.

PROPOSITION 4.2
For every interpretation $\mathcal{I} = (S, \pi, R^A)$, there exists an interpretation $\mathcal{I}' = (S, \pi, R'^A)$ such that the following two conditions hold.

1. For every $s \in S$ and for every $\varphi \in \mathcal{L}$, if $(\mathcal{I}, s) \models \varphi$, then $(\mathcal{I}', s) \models \varphi$.[6]
2. The *assumption* that *actions are completely characterized by their effects* holds in the interpretation $\mathcal{I}'$.

PROOF. To prove the proposition, we construct an interpretation $\mathcal{I}' = (S, \pi, R'^A)$ such that 'for every $\varphi \in \mathcal{L}$, if $(\mathcal{I}, s) \models [b]\varphi$, then $(\mathcal{I}, s) \models [\alpha]\varphi$' implies $(\mathcal{I}', s) \models \alpha \sqsubseteq b$.

For every action $b \in A$, let $E_{s,b}$ be the smallest set of states such that for every $\alpha \in \mathcal{A}$ for which 'for every $\varphi \in \mathcal{L}$, if $(\mathcal{I}, s) \models [b]\varphi$, then $(\mathcal{I}, s) \models [\alpha]\varphi$' holds, we have $\{t \mid (s, t) \in R^{\mathcal{A}}(\alpha)\} \subseteq E_{s,b}$. Then, we define $(s, t) \in R'^A(b)$ iff $t \in E_{s,b}$. Note that $R^A(b) \subseteq R'^A(b)$ because $b \sqsubseteq b$.

We need to check for any $\psi \in \mathcal{L}$ such that $(\mathcal{I}, s) \models [b]\psi$ whether $(\mathcal{I}', s) \models [b]\psi$ holds. The condition 'for every $\varphi \in \mathcal{L}$, if $(\mathcal{I}, s) \models [b]\varphi$, then $(\mathcal{I}, s) \models [\alpha]\varphi$' guarantees that $(\mathcal{I}', t) \models \psi$ for every $t \in E_{s,b}$. Hence, $(\mathcal{I}', s) \models [b]\psi$.

Propositions of the form $\langle b \rangle \psi$ are not affected by replacing the relation $R^A(b)$ by $R'^A(b)$. Therefore, we can prove by induction on the structure of φ that the first item of Proposition 4.2 holds.

The second item of Proposition 4.2 also holds because the construction of the interpretation $\mathcal{I}'$ guarantees that $\{t \in S' \mid (s, t) \in R'^{\mathcal{A}}(\alpha)\} \subseteq \{t \in S' \mid (s, t) \in R'^A(b)\}$. Hence, $(\mathcal{I}', s) \models \alpha \sqsubseteq b$. □

The second question mentioned above concerns propositions that hold after an action execution but that are not caused by the action. Consider, for instance, the effect axioms that describe the behaviour of atomic actions. Since they hold for abstract actions and their refinements, they have no implications on the assumption that *actions are completely characterized by their effects*.

Unfortunately, solutions to the frame problem [25] have implications on the assumption that *actions are completely characterized by their effects*. Pednault proposed a method for automatically generating frame axioms [29] for deterministic actions, which was adapted to PDL by Foo *et al.* [39].[7] This approach may introduce frame axioms that are applicable to an abstract action but not

[6] $\mathcal{I}$ and $\mathcal{I}'$ are not equivalent because $(\mathcal{I}', s) \models \alpha \sqsubseteq b$ may hold while $(\mathcal{I}, s) \not\models \alpha \sqsubseteq b$.

[7] Demolombe *et al.* [8] present a translation of Reiter's more compact solution for the frame problem [34] to an extension of PDL. PDL needs to be extended with quantification over actions.

to one of its refinements. To give an illustration, the abstract action 'going to Rome' does not say anything about the truth-value of the proposition 'arriving by train'. Therefore, Pednault's approach will introduce a frame axiom, which guarantees that the truth-value of 'arriving by train' does not change. If this proposition is false before executing the abstract action 'going to Rome', then this frame axiom ensures it is still false after executing the abstract action 'going to Rome'. Hence, the frame axiom introduces an effect of the abstract action 'going to Rome' which does not hold for the refinement 'going to Rome by train'. The same problem arises with other solutions to the frame problem. To avoid this problem, we need to know the effects of every refinement.

The cause of the problem is that we try to determine whether α is a refinement of b, which requires a solution to the frame problem, but to address the frame problem for an abstract action b, we need to know the effects of every possible refinement α. To resolve this chicken and egg problem, another approach is needed. Instead of considering all necessary effects of an action, we should focus on all necessary *causal* effects in order to determine whether α is a refinement of b. Focusing on causal effects will resolve the chicken and egg problem because the frame problem concerns non-causal effects of action executions. Below, we will address the causal effects of action executions. Firstly, however, we will address another issue.

Describing the only effects of action In dynamic logic, we cannot be sure that all necessary effects of an action a have been specified. It is always possible to add another proposition $[a]\eta$. If we could specify that a proposition φ describes the *only* necessary effects of executing a, every necessary effect of executing a must be implied by φ. Hence, we can describe all necessary effects of an action by specifying the only necessary effect of that action. This will simplify the formulation of the assumption that actions are completely characterized by their effects. For instance, if we could specify that 'being in Rome' $\wedge$ 'arriving by train' is the *only* necessary effect of the action 'going to Rome by train', then every necessary effects such as 'being in Rome' must be implied by 'being in Rome' $\wedge$ 'arriving by train'.

We propose a *new* operator $\overline{\ }$ for modifying a modal *necessity* operator $\mathcal{N}$, i.e. $\overline{\mathcal{N}}$, to denote '*only* $\mathcal{N}$'. Examples of modal necessity operators $\mathcal{N}$ are $[a]$ in case of dynamic logic, K in case of epistemic logic, B in case of doxastic logic, etc. The corresponding operators $\overline{[a]}$, $\overline{K}$ and $\overline{B}$ denote the only effects of executing a, only knowing and only believing, respectively.

DEFINITION 4.3
Let $\mathcal{N}$ be a modal necessity operator.
$\overline{\mathcal{N}}$, denoting '*only* $\mathcal{N}$', is defined as

$$(\mathcal{I}, s) \models \overline{\mathcal{N}}\varphi \text{ iff } (\mathcal{I}, s) \models \mathcal{N}\varphi \text{ and for every } \psi \in \mathcal{L}, \text{ if } (\mathcal{I}, s) \models \mathcal{N}\psi, \text{ then } \models \varphi \rightarrow \psi.$$

This definition is related to the Levesque's definition of *only-knowing* [22]. If K is the modal operator for knowing in *epistemic logic*, i.e. $K\varphi$ denotes that an agent *knows* φ, then $\overline{K}\varphi$ denotes that the agent *only knows* φ. Technically, the two definitions differ. Levesque's definition is an extension of Moore's *autoepistemic logic* [27]. Levesque specifies a semantics for Moore's autoepistemic logic using epistemic states. An epistemic state E representing an agent's knowledge is a maximal subset of the states S^*. Here, S^* is a set containing a state s for every possible truth-value assignment: $\mathcal{P} \rightarrow \{\mathsf{true}, \mathsf{false}\}$. Knowing a proposition φ given a state $s \in E$, is defined as

$$E, s \models K\varphi \text{ iff for every state } t \in S^*, \text{ if } t \in E, \text{ then } E, t \models \varphi$$

Levesque [22] shows that an epistemic state entails exactly the propositions characterized by Moore's definition [27] of an autoepistemic theory.

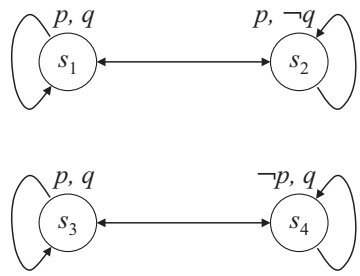


FIGURE 2 Duplicate states in an interpretation.

Since an epistemic state is a maximal set of states satisfying an agent's knowledge, it does not capture more information than that what is specified. Therefore, Levesque defines *only knowing* φ as

$E, s \models \overline{K}\varphi$ iff for every state $t \in S^*$, $t \in E$ iff $E, t \models \varphi$

Clearly, this definition implies

$$E, s \models \overline{K}\varphi \text{ iff } E, s \models K\varphi \text{ and for every } \psi \in \mathcal{L}, \text{ if } E, s \models K\psi, \text{ then } \models \varphi \rightarrow \psi.$$

The converse only holds if there are no duplicate states[8] of the states in E that do not belong to E. In *epistemic logic*, an epistemic state of an agent i is defined by an accessibility relation relative to a specific state s: $E = \{t \in S \mid (s,t) \in R_i^K\}$. If $(\mathcal{I}, s) \models \overline{K}_i\varphi$, there may exist a state u that is a duplicate of a state $v \in E = \{t \in S \mid (s,t) \in R_i^K\}$ but $u \notin E$. Duplicate states may occur in epistemic logic even in the single agent case. In standard epistemic logic, we can represent $\overline{K}p \vee \overline{K}q$ using a single interpretation. Figure 2 gives an illustration. In autoepistemic logic, $\{s_1, s_2\}$ and $\{s_3, s_4\}$ are different epistemic states.

A property of the here defined $\overline{\ }$-operator for denoting what only holds after a modal necessity operator $\mathcal{N}$ is that there always exists a proposition that characterizes 'only $\mathcal{N}$'. This proposition is unique in the sense that all propositions φ for which $\overline{\mathcal{N}}\varphi$ holds are equivalent.

PROPOSITION 4.4
Let $\mathcal{N}$ be a modal necessity-operator, let $\mathcal{I}$ be an interpretation, and let s be a state of the interpretation.

1. There exists a, possibly infinitely long, proposition ϕ such that $(\mathcal{I}, s) \models \overline{\mathcal{N}}\phi$.[9]
2. If every proposition $\mathcal{N}\psi$ can be replaced by a logically equivalent proposition $\mathcal{N}\eta$ where η is an objective proposition[10] , then there exists a proposition $\varphi \in \mathcal{L}$ of finite length such that $(\mathcal{I}, s) \models \overline{\mathcal{N}}\varphi$.[11]
3. For every $\varphi, \psi \in \mathcal{L}$, if $(\mathcal{I}, s) \models \overline{\mathcal{N}}\varphi$ and $(\mathcal{I}, s) \models \overline{\mathcal{N}}\psi$, then $\models \varphi \leftrightarrow \psi$.

[8] A state t is a duplicate of a state s if $s \neq t$ and $\pi(t) = \pi(s)$.

[9] Since $\{\psi \mid (\mathcal{I}, s) \models \mathcal{N}\psi\} \subseteq \mathcal{L}$ may contain infinitely many propositions, $\phi = \bigwedge_{(\mathcal{I},s)\models\mathcal{N}\psi} \psi$ may be of infinite length. A proposition ϕ of infinite length does not belong to $\mathcal{L}$.

[10] An objective proposition is a proposition without modal operators.

[11] This is a sufficient but not a necessary condition.

PROOF.

1. Let $\{\psi_1, \psi_2, \ldots\} \subseteq \mathcal{L}$ be the propositions for which there holds $(\mathcal{I}, s) \models \mathcal{N}\psi$, and let $\phi = \bigwedge_{\psi \in \{\psi_1, \psi_2, \ldots\}} \psi$.
   Clearly, $(\mathcal{I}, s) \models \mathcal{N}\phi$ and $\models \phi \rightarrow \psi_i$ for $i \geq 1$. Hence, $(\mathcal{I}, s) \models \overline{\mathcal{N}}\phi$.
2. Let $E = \{t \in S \mid (s, t) \in R^{\mathcal{N}}\}$. For every state $t \in E$, there is a proposition

$$\theta_t = \bigwedge_{\substack{p \in \mathcal{P} \\ \pi(t)(p) = \mathsf{true}}} p \ \wedge \bigwedge_{\substack{p \in \mathcal{P} \\ \pi(t)(p) = \mathsf{false}}} \neg p$$

   Since the number of truth-value assignment to atomic propositions is finite ($|\mathcal{P}|$ is finite) we have a finite set $\Theta = \{\theta_t \mid t \in E\}$. We define $\varphi = \bigvee \Theta$.
   Clearly, $(\mathcal{I}, s) \models \mathcal{N}\varphi$, and for every objective proposition η such that $(\mathcal{I}, s) \models \mathcal{N}\eta$, $\models \varphi \rightarrow \eta$. Hence, $(\mathcal{I}, s) \models \overline{\mathcal{N}}\varphi$.
3. Let $(\mathcal{I}, s) \models \overline{\mathcal{N}}\varphi$ and $(\mathcal{I}, s) \models \overline{\mathcal{N}}\psi$. Since $(\mathcal{I}, s) \models \overline{\mathcal{N}}\psi$, $(\mathcal{I}, s) \models \mathcal{N}\psi$, and therefore, $\models \varphi \rightarrow \psi$. Since $(\mathcal{I}, s) \models \overline{\mathcal{N}}\varphi$, $(\mathcal{I}, s) \models \mathcal{N}\varphi$, and therefore, $\models \varphi \rightarrow \psi$. Hence, $\models \varphi \leftrightarrow \phi$.

Hence, the proposition holds. □

Causal effects of actions Here, we are interested in the application of the $\overline{\ }$-operator to the necessity operator of dynamic logic. We wish to describe that the proposition 'being in Rome' describes the *only* necessary effect of the action 'going to Rome'. Unfortunately, the application of the $\overline{\ }$-operator to $[\alpha]$ raises an issue through the interaction with information that must always hold, such as effect axioms. A solution to the frame problem introduces more necessary effects, which can be difficult to determine for abstract actions; see the discussion above.

When we talk about the only necessary effect of executing an action, we actually mean the only necessary *causal* effects of executing the action. Information about unrelated actions, frame axioms, etc., is not considered. To avoid considering non-causal information when talking about what only holds after executing an action, we introduce a new modal operator to denote the causal effect φ of executing an atomic action a: $[\![a]\!]\varphi$. The causal effects φ must be an objective proposition, i.e. a proposition without modal operators. Moreover, the atomic propositions $at(\varphi)$ mentioned in φ must belong to the set $C(s, a)$ of atomic propositions that can be causally affected by the execution of the action a in the state s.

To capture the semantics of the causal effect φ of executing an atomic action a: $[\![a]\!]\varphi$, we extend an interpretation with a function $C : S \times A \rightarrow 2^{\mathcal{P}}$. So, an interpretation will be a Kripke structure of the form: $\mathcal{I} = (S, \pi, R^A, C)$. This extended interpretation enables us to define the semantics of the causal effect of executing an atomic action.

DEFINITION 4.5
Let $a \in A$ be an atomic proposition, and let φ be an objective proposition.

$(\mathcal{I}, s) \models [\![a]\!]\varphi$ iff $at(\varphi) \subseteq C(s, a)$ and $(\mathcal{I}, s) \models [a]\varphi$.

Here, the function $at(\varphi)$ denotes the set of atomic propositions occurring in the proposition φ.

Note that the definition of causal effects is syntax sensitive. This implies that two proposition φ and ψ can be equivalent while φ is a causal effect of some action and ψ is not. The only way this is possible is if ψ is the result of adding some tautology to φ. Consider for instance $l \vee \neg l$ denoting that the light is on or off. The proposition $l \vee \neg l$ can be a valid effect of the action b of pushing a button: $[\![b]\!](l \vee \neg l)$. However, adding $l \vee \neg l$ to the effect of switching on the sprinkler causing wet

grass, makes no sense: $[\![\text{sprinkler}]\!](\text{'wet grass}' \wedge (l \vee \neg l))$. The light being on or off is not a causal effect of switching on the sprinkler.

We do not define the causal effects of composite actions because the causal effects of a composite action depend on the solution of the frame problem. To give an illustration, consider an electrical circuit with two switches and a lamp. To switch on the light, we need to toggle both switches. A plan $t_1; t_2$ stating that we toggle switch 1 followed by toggling switch 2 does not have the causal effect of switching on the light without the *assumption* that the status of switch 1 does not change when toggling switch 2. In other words, the *frame problem* plays a role in describing causal effects of composite actions. A more sophisticated approach is needed to describe the causal effects of composite actions. Fortunately, for our purpose, the simple approach proposed here suffices.

The definition of the causal effect of an action is related to the necessary effect of the action.

PROPOSITION 4.6
If $(\mathcal{I}, s) \models [\![a]\!]\varphi$, then $(\mathcal{I}, s) \models [a]\varphi$.

PROOF. The result is an immediate consequence of Definition 4.5. □

The above proposition implies that we can prove a proposition for causal effects similar to Proposition 3.3.

PROPOSITION 4.7
For every interpretation state pair $(\mathcal{I}, s)$, if $(\mathcal{I}, s) \models \alpha \sqsubseteq b$, then for every $\varphi \in \mathcal{L}$, if $(\mathcal{I}, s) \models [\![b]\!]\varphi$, then $(\mathcal{I}, s) \models [\alpha]\varphi$.

PROOF. The proposition directly follows from the previous proposition and Proposition 3.3. □

Several proposals have been made to address causality in PDL. $\mathcal{LAP}_{\rightsquigarrow}$ [4] is a simplified version of PDL which allows for the explicit specification that the truth-value of a literal is causally affected by an action execution. This specification is used to address the frame problem by specifying that the truth-value of a literal ℓ is independent of executing an action a: $a \not\rightsquigarrow \ell$. We did not choose this approach here because we needed a modal operator specifying the causal effect of an action execution. Of course, we could have encoded $[\![a]\!]\varphi$ in terms of $\mathcal{LAP}_{\rightsquigarrow}$:

$(\mathcal{I}, s) \models [\![a]\!]\varphi$ iff $(\mathcal{I}, s) \models [a]\varphi$ and for every $p \in at(\varphi)$, $(\mathcal{I}, s) \models a \rightsquigarrow p$ and $(\mathcal{I}, s) \models a \rightsquigarrow \neg p$.

Zhang and Foo [38] propose an extended PDL (EPDL), which interprets standard modal operators $\langle a \rangle$ and $[a]$ of PDL as operators for describing causal effects. EPDL is not really an extension of PDL because we no longer have the standard modal operators for possible and necessary effects of actions. Since we wish to stay in the domain of standard dynamic logic, we did not consider EPDL for our purpose. The same as our solution, EPDL does not provide a solution for the causal effect of composite actions. The authors do point out that frame axioms need to be considered. To address the frame problem, we need to describe the 'non-causal effects' of action executions. Since the modal operators $\langle a \rangle$ and $[a]$ are used to describe causal effects, they can no longer be used to describe 'non-causal effects'.

Deriving the refinement relation from the effects of actions Applying the $\overline{\ }$-operator to $[\![a]\!]$ enables us to describe the *only* causal effect of an atomic action a.

$(\mathcal{I}, s) \models \overline{[\![a]\!]}\varphi$ iff $(\mathcal{I}, s) \models [\![a]\!]\varphi$ and for every $\psi \in \mathcal{L}$, if $(\mathcal{I}, s) \models [\![a]\!]\psi$, then $\models \varphi \rightarrow \psi$.

Note that item 2 of Proposition 4.4 remains valid if we restrict the set of atomic proposition to $C(s, a)$.

Since $\overline{[\![a]\!]}\varphi$ describes the only causal effects, we might expect that $at(\varphi) = C(s, a)$. The current semantics does not enforce this. It is always possible to add a proposition $\overline{[\![a]\!]}(\varphi \wedge (p \vee \neg p))$ where $p \notin at(\varphi)$.

The operator $\overline{[\![a]\!]}$ for describing the only causal effect of an action provides a way to identify refinement relations between actions. Firstly, we need to reformulate our assumption that actions are completely characterized by their effects. Ideally, we should reformulate the assumption in terms of the causal effect of the abstract and the refined action. Since the latter can be a composite action, we will consider the necessary effect instead of the causal effect of the refined action. As a consequence the plan $(\neg rain?;\ wait)^*;\ rain?$ will be a refinement of the abstract action of *make_it_rain*. The plan is not a proper refinement because rainfall is not a causal effect of the plan. Using the necessary effect of the refined action is not a problem in environments where changes are always the result of the executed actions.

DEFINITION 4.8
We say that *actions are completely characterized by their* causal *effects* in a state s of an interpretation $\mathcal{I}$ iff the following condition holds:

$(\mathcal{I}, s) \models \alpha \sqsubseteq b$ iff for every$\varphi \in \mathcal{L}$, if$(\mathcal{I}, s) \models [\![b]\!]\varphi$, then$(\mathcal{I}, s) \models [\alpha]\varphi$.

The assumption that actions are completely characterized by their *causal* effects is equivalent to a property that is more useful to identify refinement relations between actions. If we know the only necessary causal effect of an atomic action b, we can determine that a more specific (composite) action α is a refinement of the action b.

PROPOSITION 4.9
The assumption:

$(\mathcal{I}, s) \models a \sqsubseteq b$ iff for every$\varphi \in \mathcal{L}$, if$(\mathcal{I}, s) \models [\![b]\!]\varphi$, then$(\mathcal{I}, s) \models [\alpha]\varphi$

is equivalent to

$(\mathcal{I}, s) \models \alpha \sqsubseteq b$ iff there is a $\varphi \in \mathcal{L}$ such that$(\mathcal{I}, s) \models \overline{[\![b]\!]}\varphi$ and $(\mathcal{I}, s) \models [\alpha]\varphi$

PROOF. It is sufficient to prove that 'there is a $\varphi \in \mathcal{L}$ such that $(\mathcal{I}, s) \models \overline{[\![b]\!]}\varphi$ and $(\mathcal{I}, s) \models [\alpha]\varphi$' is equivalent to 'for every $\varphi \in \mathcal{L}$, if $(\mathcal{I}, s) \models [b]\varphi$, then $(\mathcal{I}, s) \models [\alpha]\varphi$'.

($\Rightarrow$) Suppose there is a $\varphi \in \mathcal{L}$ such that $(\mathcal{I}, s) \models \overline{[\![b]\!]}\varphi$ and $(\mathcal{I}, s) \models [\alpha]\varphi$. Moreover, suppose that for some $\psi \in \mathcal{L}$, $(\mathcal{I}, s) \models [\![b]\!]\psi$ and $(\mathcal{I}, s) \not\models [\alpha]\psi$. Since $(\mathcal{I}, s) \models \overline{[\![b]\!]}\varphi$, $\models \varphi \rightarrow \psi$. Therefore, $(\mathcal{I}, s) \models [\alpha]\psi$. Contradiction. Hence, for every $\varphi \in \mathcal{L}$, if $(\mathcal{I}, s) \models [\![b]\!]\varphi$, then $(\mathcal{I}, s) \models [\alpha]\varphi$.

($\Leftarrow$) Suppose that for every $\varphi \in \mathcal{L}$, if $(\mathcal{I}, s) \models [\![b]\!]\varphi$, then $(\mathcal{I}, s) \models [\alpha]\varphi$. According to the second item of Proposition 4.4, there is a $\psi \in \mathcal{L}$ such that $(\mathcal{I}, s) \models \overline{[\![b]\!]}\psi$. Since $(\mathcal{I}, s) \models [\![b]\!]\psi$, also $(\mathcal{I}, s) \models [\alpha]\psi$ must hold. □

The above result implies that it is no longer necessary to specify all possible refinements of abstract actions in the initial information. Instead, it suffices to specify the only causal effects of abstract actions. Of course, knowing the refinement relation over actions in advance, makes planning more efficient.

A last point that we need to address concerns the necessary effects of an abstract action. Necessary effects of an abstract action must also be necessary effects of each refinement. Since we identify a

refinement using the only *causal* effect instead of the only *necessary* effects, a necessary effect of an abstract action a priori need not be necessary effect of a refinement. The reasoning process must ensure that necessary effects of an abstract action are also necessary effects of each refinement.

Achievement Zhou and Zhang [40] propose to add an *achievement operator* to PDL. The achievement of a proposition φ, denoted by $\#\varphi$, is the most abstract action resulting in φ to hold. The semantics that Zhou and Zhang propose for the operator is every transition to a state in which φ holds: $R^{\mathcal{A}}(\#\varphi) = \{(s,t) \mid s \in S, t \in S, (\mathcal{I},t) \models \varphi\}$. There are two problems with this semantic definition of achievement. The first problem is the lack of a causal relation. Achieving φ means that the abstract action $\#\varphi$ must *cause* φ to hold; otherwise, it is just a coincidence. The second problem is the interaction with other information. Suppose that we have the information $\Box p \oplus \Box q$ where $\oplus$ denotes the exclusive-or. Then $R^{\mathcal{A}}(\#r)$ should not contain a couple (s,t) where $(\mathcal{I},s) \models \Box p$, $(\mathcal{I},t) \models \Box q$ and $(\mathcal{I},s) \models r$.

The proposition φ should be a causal effect of the achievement of φ. Therefore, we can specify the achievement of φ by $\overline{[\![\#\varphi]\!]}\varphi$. Since this should always hold in every state, we get $\models \Box\overline{[\![\#\varphi]\!]}\varphi$. This characterization of the achievement of φ implies the following semantics specification of the achievement of φ:

- $R^{\mathcal{A}}(\#\varphi) = \{(s,t) \mid (s,t) \in \left(\bigcup_{a\in A} R^A(a)\right)^*, (\mathcal{I},t) \models \varphi\}$;
- for every $s \in S$, if $(\mathcal{I},s) \models [\![\#\varphi]\!]\psi$, then $\models \varphi \rightarrow \psi$;
- $C(s,\#\varphi) = at(\varphi)$ for every $s \in S$.

Of course, a composite action that realizes the achievement should have φ as a causal effect. As pointed out above, determining the causal effect of a composite action requires a solution for frame problem.

5 Proof system

In the previous two sections, we have extended dynamic logic with an operator for specifying a refinement relation over actions. We did not yet address how to reason with a refinement relation over actions. Reasoning with the refinement relation will be the focus of this section.

5.1 Preliminaries

The reasoning process is based on a semantic tableau method, specifically, the prefixed-tableau method. We will extend the prefixed-tableau described by De Giacomo and Massacci [14].

Generally, a tableau rule is of the form

$$\frac{x_1 : \varphi_1, \ldots, x_m : \varphi_k}{y_{1,1} : \psi_{1,1}, \ldots, y_{1,k_1} : \psi_{1,k_1} \mid \ldots \mid y_{n,1} : \psi_{n,1}, \ldots, y_{n,k_n} : \psi_{n,k_n}}$$

Here, φ_i and $\psi_{i,j}$ denote propositions, and x_i and $y_{i,j}$ denote prefixes. A prefix can be viewed as encoding a path in an interpretation. The prefix $0.\langle a\rangle.3.\langle b\rangle.9.\langle c\rangle.11$ denotes the states: s_0, s_3, s_9 and s_{11}, and instances (s_0,s_3), (s_3,s_9) and (s_9,s_{11}) of the accessibility relations $R^A(a)$, $R^A(b)$ and $R^A(c)$, respectively. In the last state s_{11} of the prefix, the proposition after colon holds. Below, we will also use some non-standard rules containing a test in the precondition of the tableau rule or a condition in the consequent. The condition in the consequent must be evaluated separately.

We will use s^x to denote the last state s_n specified by a prefix $x = y.\langle a\rangle.n$ with $n \in \mathbb{N}$, $a \in A$ and y being a prefix. This avoids the need to talk about the structure of the prefix x. We will use $x = y.z$ to

denote that the prefix x is the result of extending the prefix y with several additional state transitions and states, denoted by z.

To prove the *correctness* of a semantic tableau method, we need to prove that one of the leafs is satisfiable if the root of the tableau is satisfiable. We can prove this in two steps. Firstly, we prove for every rule application that one of its child nodes is satisfiable if the parent node on which the rule is applied, is satisfiable. Next, we prove with induction on construction of the tableaux the correctness of the semantic tableau method. De Giacomo and Massacci [14] proved the correctness of the prefixed-tableau for the tableau rules described in Section 2. To extend this proof for the new rules described below, it is sufficient to prove the correctness of the new rules.

To prove the completeness of a semantic tableaux method, we need to show that there exists an interpretation entailing all the propositions of an open leaf of the semantic tableau. Since the root is a subset of each leaf of the tableau, the root is satisfiable if there is an open leaf. For dynamic logic without refinement relation, we can prove that the following interpretation satisfies all propositions in a saturated open leaf Λ of the tableau.

Let $\mathcal{I} = (S, \pi, R^A)$ be an interpretation constructed for a saturated open leaf Λ. This interpretation is defined as follows:

- $S = \{s^x \mid x : \varphi \in \Lambda\}$
- $R^A(a) = \{(s^z, s_n) \mid x : \varphi \in \Lambda, x = z.\langle a\rangle.n, x \text{ is not blocked}\} \cup$
  $\{(s^z, s^y) \mid x : \varphi \in \Lambda, x = z.\langle a\rangle.n, x \text{ is directly blocked by } y\}$
- $\pi(s^x)(p) =$ true if $x : p \in \Lambda$ and $p \in \mathcal{P}$; false otherwise

We prove that there also exists an interpretation satisfying an open leaf when considering the new operators introduced in the previous sections. The proof for the new operators extends the completeness proof of PDL and may require extending the above interpretation. Since the proof uses induction on the structure of the proposition, we refer to the proof of the sub-proposition as the *induction hypothesis*.

5.2 The refinement relation

The tableau rules for the refinement relation express the implications of the refinement relation.[12]

$$\frac{x{:}\alpha \sqsubseteq b, x{:}[b]\varphi}{x{:}[\alpha]\varphi} \qquad \frac{x{:}\neg(\alpha \sqsubseteq \beta)}{x{:}\langle\alpha\rangle\xi, x{:}\neg\langle\beta\rangle\xi}$$

Note that we do not introduce a rule $\frac{x{:}\alpha \sqsubseteq b, x{:}\langle\alpha\rangle\varphi}{x{:}\langle b\rangle\varphi}$. Although $\Gamma \models \langle\alpha\rangle\varphi$ may hold for a node Γ of a tableau, no descendant of Γ may contain $x : \langle\alpha\rangle\varphi$. Moreover, for the correctness and the completeness of the semantic tableau method, this rule is not required because it is equivalent to the left tableau rule.

The proposition $\xi \in \Xi$ in the second rule must be a *new* atomic proposition that is a *witness* of a state. Unlike nominals in hybrid logic [1], there is no need to introduce special tableau rules for handling witnesses because witnesses can only be introduced by the above rule. Since we do not know the state denoted by the proposition ξ, the second rule may not seem to be very useful. Nevertheless, it can be used to derive new conclusions as is shown by the following example.

[12] If the right-hand side of the refinement relation can be a composite action instead of an atomic abstract action, then the rule in the first column should be replaced by $\frac{x{:}\alpha \sqsubseteq \beta}{x{:}[\alpha]\psi, x{:}\langle\beta\rangle\neg\psi}$, where ψ can be any proposition in $\mathcal{L}$. We might use information in Γ to restrict the propositions ψ to be considered.

EXAMPLE 5.1
We prove that $\{a \sqsubseteq b, b \sqsubseteq c\} \models a \sqsubseteq c$ by constructing the following semantic tableau:

$$
\begin{array}{c}
0 : a \sqsubseteq b \\
0 : b \sqsubseteq c \\
\boxed{0 : \neg(a \sqsubseteq c)} \\
| \\
0 : \langle a \rangle \xi \\
0 : \neg\langle c \rangle \xi \\
| \\
0.\langle a \rangle .1 : \xi \\
| \\
0 : [c]\neg\xi \\
| \\
0 : [b]\neg\xi \\
| \\
0 : [a]\neg\xi \\
| \\
0.\langle a \rangle .1 : \neg\xi \\
\times
\end{array}
$$

In the above tableau, we firstly rewrite $0 : \neg(a \sqsubseteq b)$ using the rule $\frac{x:\neg(\alpha \sqsubseteq \beta)}{x:\langle\alpha\rangle\xi, x:\neg\langle\beta\rangle\xi}$. Next, we rewrite $0 : \langle a \rangle \xi$ and $0 : \neg\langle c \rangle \xi$ using the tableau rules for PDL. Subsequently, we rewrite $0 : b \sqsubseteq c$, $0 : [c]\neg\xi$, followed by rewriting $0 : a \sqsubseteq b$, $0 : [b]\neg\xi$ using the rule $\frac{x:\alpha \sqsubseteq b, x:[b]\varphi}{x:[\alpha]\varphi}$. Finally, we rewrite $0 : [a]\neg\xi$ using a tableau rule for PDL. The tableau closes because the branch contains $0.\langle a \rangle .1 : \xi$ and $0.\langle a \rangle .1 : \neg\xi$.

Note that the addition of the refinement relation with the restriction that only atomic propositions can be refined does not make the logic undecidable. Whether this also holds without the restriction that only atomic propositions can be refined, as is claimed by Herzig *et al.* [18], is unclear.

Correctness and completeness of tableau rules for the refinement relations The rules relating the refinement relation between actions to dynamic logic are valid tableau rules.

LEMMA 5.2
Let Γ be a node of the semantic tableau, and let Γ' be the directly succeeding node that is the result of applying one of the two tableau rules for the refinement relation.

If Γ is satisfiable, then Γ' is satisfiable.[13]

PROOF. Propositions 3.3 and 3.2 imply the correctness of the lemma for the left and the right rules, respectively. □

For the completeness proof of the reasoning process, we need to be able to construct an interpretation for an open leaf of the tableau. We will adapt the interpretation $\mathcal{I} = (S, \pi, R^A)$ for a saturated open leaf Λ described in Subsection 5.1. The adaptation consists of extending $R^A(b)$.

[13] As defined in Section 2, we use Γ to denote a node of a tableau as well as the set of propositions represented by the node.

- For every action $b \in A$, let $E_{s,b}$ be the smallest set of states such that for every $\alpha \in \mathcal{A}$ for which 'for every $\varphi \in \mathcal{L}$, if $(\mathcal{I}, s) \models [b]\varphi$, then $(\mathcal{I}, s) \models [\alpha]\varphi$' holds, we have $\{t \mid (s,t) \in R^{\mathcal{A}}(\alpha)\} \subseteq E_{s,b}$. Then, we define $(s,t) \in R'^{A}(b)$ iff $t \in E_{s,b}$. Note that $R^{A}(b) \subseteq R'^{A}(b)$ because $b \sqsubseteq b$.

The following lemma shows that the adapted interpretation entails Λ.

LEMMA 5.3
Let Λ be a saturated open leaf of a semantic tableau.
Then Λ is satisfiable.

PROOF. We prove the lemma by showing that the adapted interpretation $\mathcal{I}$ described above, entails Λ. Firstly, we address the entailment of $\alpha \sqsubseteq b$ and of $\neg(\alpha \sqsubseteq b)$.

- Let $x : \alpha \sqsubseteq b \in \Lambda$, $\alpha \in \mathcal{A}$ and $b \in A$. The rule $\frac{x:\alpha \sqsubseteq b, x:[b]\varphi}{x:[\alpha]\varphi}$ ensures that $x : [\alpha]\varphi \in \Lambda$ for every $x : [b]\varphi \in \Lambda$. This implies that the condition 'for every $\varphi \in \mathcal{L}$, if $(\mathcal{I}, s^x) \models [b]\varphi$, then $(\mathcal{I}, s^x) \models [\alpha]\varphi$' mentioned in the above definition of the interpretation $\mathcal{I}$ holds. Therefore, by the construction of $\mathcal{I}$, $\{t \mid (s^x, t) \in R^{\mathcal{A}}(\alpha)\} \subseteq \{t \mid (s^x, t) \in R^{A}(b)\}$. Hence, $(\mathcal{I}, s^x) \models \alpha \sqsubseteq b$.
- Let $x : \neg(\alpha \sqsubseteq b) \in \Lambda$, $\alpha \in \mathcal{A}$ and $b \in A$. Then the rule $\frac{x:\neg(\alpha \sqsubseteq b)}{x:\langle\alpha\rangle\xi, x:\neg\langle b\rangle\xi}$ must have been applied. So, according to the induction hypothesis, there is a state $t \in S$ such that $(s^x, t) \in R^{\mathcal{A}}(\alpha)$, $(\mathcal{I}, t) \models \xi$ and $(s^x, t) \notin R^{A}(b)$. Hence, $(\mathcal{I}, s^x) \models \neg(\alpha \sqsubseteq b)$.

The entailment of $[b]\varphi$ has to be readdressed also because the above described modification has changed the relation $R^{A}(b)$.

- Let $x : [b]\varphi \in \Lambda$ and $b \in A$.
  Suppose that $(\mathcal{I}, s^x) \not\models [b]\varphi$. Then there is a state t such that $(s^x, t) \in R^{A}(b)$ and $(\mathcal{I}, t) \not\models \varphi$. Let y be the prefix denoting the state t. According to the definition of $R^{A}(b)$, $y = x.\langle b\rangle.n$, or $z = x.\langle b\rangle.n$ and y directly blocks z, or $y = x.u$ where u encodes the execution of the composite action α, $x : \alpha \sqsubseteq b \in \Lambda$, and $R^{A}(b)$ has been updated with instances of the relation $R^{\mathcal{A}}(\alpha)$. The first two cases are addressed in the proof that $\mathcal{I}$ entails Λ. The third case is new and will be described below.
  - In the third case, since $\{x : \alpha \sqsubseteq b, x : [b]\varphi\} \subseteq \Lambda$, the rule $\frac{x:\alpha \sqsubseteq b, x:[b]\varphi}{x:[\alpha]\varphi}$ must have been applied. Hence, according to the induction hypothesis, for every state $t' \in S$, if $(s^x, t') \in R^{\mathcal{A}}(\alpha)$, then $(\mathcal{I}, t') \models \varphi$. Therefore, $(\mathcal{I}, t) \models \varphi$. Contradiction.

  Hence, $(\mathcal{I}, s^x) \models [b]\varphi$.

For the other proposition in Λ, nothing has changed, and therefore, the original proof still applies. □

5.3 All necessary effects

We have introduced a $\overline{\ }$-operator for specifying all effects of a modal necessity operator $\mathcal{N}$.[14] To reason with propositions containing $\overline{\mathcal{N}}$, we need the following tableau rules. In the right rule, $\boldsymbol{\eta}$ must be a new atomic proposition not belonging to the set of atomic propositions $\mathcal{P}$ that are used to define the language $\mathcal{L}$.

$$\frac{x:\overline{\mathcal{N}}\varphi}{x:\mathcal{N}\varphi} \qquad \frac{x:\overline{\mathcal{N}}\varphi, x:\mathcal{N}\psi}{\models(\varphi \rightarrow \psi)} \qquad \frac{x:\neg\overline{\mathcal{N}}\varphi}{x:\neg\mathcal{N}\varphi \mid x:\mathcal{N}\boldsymbol{\eta}, \not\models(\varphi \rightarrow \boldsymbol{\eta})}$$

[14] In case of dynamic logic $\mathcal{N}$ should be replaced by $[\alpha]$.

The condition $\models (\varphi \rightarrow \psi)$ in the consequence of the middle rule must be evaluated in a separate semantic tableau. In this tableau, we assume that $0 : \neg(\varphi \rightarrow \psi)$ holds and then try to close it. The reason why we need a separate tableau is because we are trying to determine an interpretation-state pair $(\mathcal{I}, s)$ entailing the initial information in the root of the tableau while $(\varphi \rightarrow \psi)$ must hold for every $(\mathcal{I}', s')$. So, the branch containing $\models (\varphi \rightarrow \psi)$ *closes* if a tableau with root $0 : \neg(\varphi \rightarrow \psi)$ *does not close*.

EXAMPLE 5.4
We prove that $\overline{[a]}(p \vee q) \models \neg[a]p$ by constructing the following semantic tableau:

$$\begin{array}{c} 0 : \overline{[a]}(p \vee q) \\ \boxed{0 : \neg(\neg[a]p)} \\ | \\ 0 : [a]p \\ | \\ \models (p \vee q) \rightarrow p \\ | \\ ? \end{array}$$

We apply the middle rule given $0 : \overline{[a]}(p \vee q)$ and $0 : [a]p$. To decide whether the tableau closes, we investigate in a separate tableau whether $\models (p \vee q) \rightarrow p$ holds.

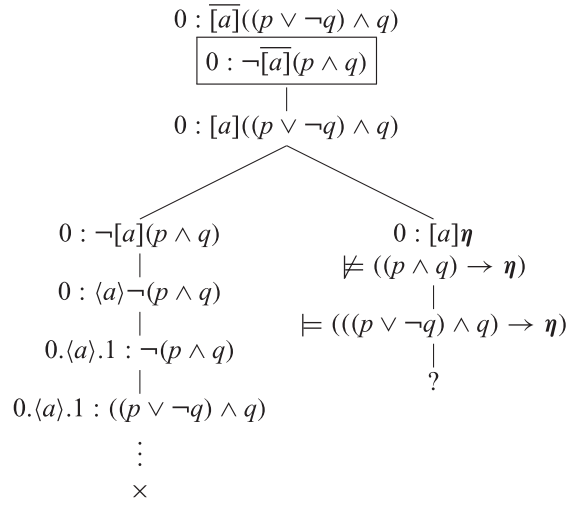
$$\begin{array}{c} 0 : \neg((p \vee q) \rightarrow p) \\ | \\ 0 : p \vee q \\ 0 : \neg p \\ \diagup \quad \diagdown \\ 0 : p \qquad 0 : q \\ | \\ \times \end{array}$$

This tableau does not close. Therefore, $\not\models (p \vee q) \rightarrow p$, and the branch of the first tableau closes.

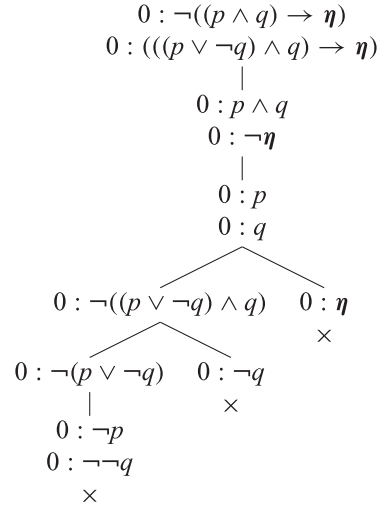
In the right most rule, $\boldsymbol{\eta}$ denotes *some* proposition in $\mathcal{L}$, i.e. we can view $\boldsymbol{\eta}$ as an existentially quantified propositional variable. Since we do not know which proposition in $\mathcal{L}$ is represented by $\boldsymbol{\eta}$, we may treat $\not\models (\varphi \rightarrow \boldsymbol{\eta})$ as a *constraint* on the possible choices of the proposition $\boldsymbol{\eta} \in \mathcal{L}$. This constraint may be used to evaluate the condition in the consequent of the middle rule. To give an illustration, assume that we have a node Γ with $\{x : \overline{\mathcal{N}}\lambda, x : \neg\overline{\mathcal{N}}\mu\} \subseteq \Gamma$. Applying the right rule gives us $x : \mathcal{N}\boldsymbol{\eta}$ and the constraint $\not\models (\mu \rightarrow \boldsymbol{\eta})$ in the right branch. The constraint $\not\models (\mu \rightarrow \boldsymbol{\eta})$ implies that there exists an interpretation-state pair $(\mathcal{I}, s)$ such that $(\mathcal{I}, s) \not\models (\mu \rightarrow \boldsymbol{\eta})$. If we subsequently apply the middle rule in the right branch, the consequent $\models (\lambda \rightarrow \boldsymbol{\eta})$ will be added to the next node of the branch. The consequence $\models (\lambda \rightarrow \boldsymbol{\eta})$ states that for every interpretation $\mathcal{I}'$, $\mathcal{I}' \models (\lambda \rightarrow \boldsymbol{\eta})$. So, $(\mathcal{I}, s) \models (\lambda \rightarrow \boldsymbol{\eta})$. If the semantic tableau with the root $\{0 : \neg(\mu \rightarrow \boldsymbol{\eta}), 0 : \lambda \rightarrow \boldsymbol{\eta}\}$ closes,[15] then $(\mathcal{I}, s) \not\models (\lambda \rightarrow \boldsymbol{\eta})$, and therefore, $\mathcal{I} \not\models (\lambda \rightarrow \boldsymbol{\eta})$. Hence, $\not\models (\lambda \rightarrow \boldsymbol{\eta})$ and the branch on which we apply the middle rule, closes. The following example gives a practical application.

[15] The prefix 0 denotes the state s of the interpretation $\mathcal{I}$.

EXAMPLE 5.5
We prove that $\overline{[a]}((p \vee \neg q) \wedge q) \models \overline{[a]}(p \wedge q)$ by constructing the following semantic tableau:



The instantiated rule $\frac{o:\overline{[a]}((p\vee\neg q)\wedge q)}{x:[a]((p\vee\neg q)\wedge q)}$ is applied before the split into two branches. Next, the instantiated rule $\frac{0:\neg\overline{[a]}(p\wedge q)}{0:\neg[a]((p\vee\neg q)\wedge q)|0:[a]\boldsymbol{\eta},\not\models(p\wedge q)\rightarrow\boldsymbol{\eta})}$ is applied resulting in the split into two branches. The left branch can be closed by applying the standard tableau rules for PDL. The instantiated rule $\frac{0:\overline{[a]}((p\vee\neg q)\wedge q),x:[a]\boldsymbol{\eta}}{\models(((p\vee\neg q)\wedge q)\rightarrow\boldsymbol{\eta})}$ is applied in the right branch. No other rule can be applied in the right branch. The only thing we can do is evaluating whether $\models (((p \vee \neg q) \wedge q) \rightarrow \boldsymbol{\eta})$ given the constraint $\not\models ((p \wedge q) \rightarrow \boldsymbol{\eta})$. A separate tableau is used for this evaluation.



This tableau closes. Therefore, $\not\models (((p\vee\neg q)\wedge q) \rightarrow \eta)$, and the rightmost branch of the first tableau closes.

Correctness and completeness of tableau rules for all necessary effects The following two lemmas addresses the validity of the tableau rules for necessary effects.

LEMMA 5.6
Let Γ be a node of the semantic tableau, and let Γ' and possibly Γ'' be the directly succeeding nodes that are the result of applying one of the three tableau rules listed above.

If Γ is satisfiable, then Γ' or Γ'' is satisfiable.

PROOF. The lemma directly follows from the definition of the $\overline{\ }$-operator. □

For the completeness proof of the reasoning process, we need to be able to construct an interpretation for a saturated open leaf of the tableau. We will adapt the interpretation $\mathcal{I} = (S, \pi, \ldots, R^{\mathcal{N}}, \ldots)$ with accessibility relation $R^{\mathcal{N}}$, for a saturated open leaf Λ. The underlying reason for the adaptation is the use of the rule $\frac{x:\overline{\mathcal{N}}\varphi, x:\mathcal{N}\psi}{\models(\varphi\rightarrow\psi)}$ instead of the rule $\frac{x:\overline{\mathcal{N}}\varphi}{x:\neg\mathcal{N}\psi \mid \models(\varphi\rightarrow\psi)}$. The latter rule must be applied for every $\psi \in \mathcal{L}$, which is practically infeasible.

We make the following adaptation.

- For every $x : \overline{\mathcal{N}}\varphi \in \Lambda$ and for every $\mu \in \mathcal{L}$ such that $(\mathcal{I}, s^x) \models \mathcal{N}\mu$ and $\not\models \varphi \rightarrow \mu$, we add a new state s_n to S, a new couple (s^x, s_n) to the relation $R^{\mathcal{N}}$, and we ensure that $(\mathcal{I}', s_n) \models \neg\mu \wedge \varphi$ holds.

The resulting interpretation is denoted by $\mathcal{I}'$. Note that we can make this adaptation because in a saturated leaf Λ, the second rule has been applied for every $x : \mathcal{N}\psi \in \Lambda$. Therefore, for every $x : \mathcal{N}\psi \in \Lambda$, $\models \varphi \rightarrow \psi$. Since $\not\models \varphi \rightarrow \mu$, $\{\psi \mid x : \mathcal{N}\psi \in \Lambda\} \not\models \mu$.

LEMMA 5.7
Let Λ be the set of propositions of a *saturated* open leaf of the semantic tableau.

Then there exists an interpretation $\mathcal{I}'$ entailing Λ, especially entailing every $x : \overline{\mathcal{N}}\varphi \in \Lambda$.

PROOF. We prove the lemma by showing that the adapted interpretation $\mathcal{I}'$ described above, entails Λ. Firstly, we address the entailment of $\overline{\mathcal{N}}\varphi$ and of $\neg\overline{\mathcal{N}}\varphi$.

- Let $x : \overline{\mathcal{N}}\varphi \in \Lambda$. Suppose that $(\mathcal{I}', s^x) \not\models \overline{\mathcal{N}}\varphi$. Since $x : \mathcal{N}\varphi \in \Lambda$, according to the induction hypothesis, $(\mathcal{I}', s^x) \models \mathcal{N}\varphi$. Therefore, there is a proposition η such that $(\mathcal{I}', s^x) \models \mathcal{N}\eta$ but not $\models \varphi \rightarrow \eta$. Clearly, also for the original interpretation $\mathcal{I}$, $(\mathcal{I}, s^x) \models \mathcal{N}\eta$. But then the adapted interpretation $\mathcal{I}'$ must contain a state s_n such that $(s^x, s_n) \in R^{\mathcal{N}}$ and $(\mathcal{I}', s_n) \models \neg\eta \wedge \varphi$. Hence, $(\mathcal{I}', s^x) \not\models \mathcal{N}\eta$. Contradiction.
  Hence, $(\mathcal{I}', s^x) \models \overline{\mathcal{N}}\varphi$.
- Let $x : \neg\overline{\mathcal{N}}\varphi \in \Lambda$. Suppose that $x : \neg\mathcal{N}\varphi \in \Lambda$. Then, according to the induction hypothesis, $(\mathcal{I}', s^x) \models \neg\mathcal{N}\varphi$, and therefore, $(\mathcal{I}', s^x) \models \neg\overline{\mathcal{N}}\varphi$.
  Suppose that for some proposition $\eta \in \mathcal{L}$, $x : \mathcal{N}\eta \in \Lambda$ and $\not\models \varphi \rightarrow \eta$. Then, according to the induction hypothesis, $(\mathcal{I}', s^x) \models \mathcal{N}\eta$ and $\not\models \varphi \rightarrow \eta$. Therefore, according to the definition of the $\overline{\ }$-operator, $(\mathcal{I}', s^x) \models \neg\overline{\mathcal{N}}\varphi$.

For the other propositions in Λ, nothing has changed, and therefore, the original proof still applies. □

5.4 Causal effects

Since causal effects are necessary effects, we have the following tableau rules.

$$\frac{x:[\![a]\!]\varphi}{x:[a]\varphi, C(x,a) := C(x,a) \cup at(\varphi)} \qquad \frac{x:\neg[\![a]\!]\varphi, at(\varphi) \subseteq C(x,a)}{x:\neg[a]\varphi}$$

The set $C(x, a)$ is used to keep track of the atomic propositions that are causally affected by the atomic action a in the state s^x denoted by the prefix x. For each new prefix x, $C(x, a)$ is equal to the empty set and is updated by applying the left rule.

The right rule is a reformulation of $\frac{x:\neg[\![a]\!]\varphi}{x:\neg[a]\varphi | at(\varphi) \not\subseteq C(x,a)}$, which is implied by the definition of the causal effect of an atomic action; Definition 4.5. The condition $at(\varphi) \not\subseteq C(x, a)$ is a *constraint* that must be rechecked each time the left rule is applied. This is avoided by the reformulated right rule.

EXAMPLE 5.8

We prove that $[\![a]\!](p \wedge q) \models [\![a]\!]p$ by constructing the following semantic tableau:

$$
\begin{array}{c}
0 : [\![a]\!](p \wedge q) \\
\boxed{0 : \neg[\![a]\!]p} \\
| \\
0 : [a](p \wedge q) \\
C(0, a) := \varnothing \cup \{p, q\} \\
| \\
0 : \neg[a]p \\
| \\
0 : \langle a \rangle \neg p \\
| \\
0.\langle a \rangle.1 : \neg p \\
| \\
0.\langle a \rangle.1 : p \\
| \\
\times
\end{array}
$$

We first apply the left rule, which makes it possible to next apply the right rule. The remaining steps are the result of applying standard PDL tableau rules.

We *cannot* prove that $[\![a]\!]p \models [\![a]\!](p \vee q)$ as is shown by the following tableau:

$$
\begin{array}{c}
0 : [\![a]\!]p \\
\boxed{0 : \neg[\![a]\!](p \vee q)} \\
| \\
0 : [a]p \\
C(0, a) := \varnothing \cup \{p\}
\end{array}
$$

We can only apply the left rule but no other tableau rules. The right rule *cannot* be applied because $at(p \vee q) = \{p, q\} \not\subseteq \{p\} = C(0, a)$. Note that the tableau closes if we could apply the right rule.

Correctness and completeness of tableau rule for causal effects The following lemma addresses the validity of the tableau rules.

LEMMA 5.9

Let Γ be a node of the semantic tableau, and let Γ' be the directly succeeding node that is the result of applying one of the two above listed tableaux rules.

If Γ is satisfiable, then Γ' is satisfiable.

PROOF. The correctness of the lemma is a direct consequence of Definition 4.5. □

The following lemma shows that an interpretation can be constructed for an open leaf.

LEMMA 5.10
Let Λ be the set of propositions of a *saturated* open leaf of the semantic tableau.

Then there exists an interpretation $\mathcal{I} = (S, \pi, R^A, C)$ entailing Λ.

PROOF. Let $x : [\![a]\!]\varphi \in \Lambda$. Then, $x : [a]\varphi \in \Lambda$ and $at(\varphi) \subseteq C(x, a)$. Therefore, for every $(s^x, t) \in R^A(a)$, $(\mathcal{I}, t) \models \varphi$ and $at(\varphi) \subseteq C(s^x, a)$. Hence, $(\mathcal{I}, s^x) \models [\![a]\!]\varphi$.

Let $x : \neg[\![a]\!]\varphi \in \Lambda$.

- Suppose that $at(\varphi) \not\subseteq C(x, a)$. Then $at(\varphi) \not\subseteq C(s^x, a)$, and therefore, $(\mathcal{I}, s^x) \models \neg[\![a]\!]\varphi$.
- Suppose that $at(\varphi) \subseteq C(x, a)$. Then $x : \neg[a]\varphi \in \Lambda$, $at(\varphi) \subseteq C(s^x, a)$ and $(\mathcal{I}, s^x) \models \neg[a]\varphi$. Hence, $(\mathcal{I}, s^x) \models \neg[\![a]\!]\varphi$.

□

5.5 Characterize actions by their effects

The tableau rule $\frac{x:\neg(a \sqsubseteq b)}{x:\langle a\rangle\xi, x:\neg\langle b\rangle\xi}$ introduced in Subsection 5.2 cannot always be used to derive a refinement relation over actions using a refutation proof because we do not know the implications of the witness proposition ξ characterizing a state. Actions that are completely characterized by their effects, offer an alternative way of deriving a refinement relation. As was argued in Section 4, we should only consider *causal* effects.

$$\frac{x:\neg(\alpha \sqsubseteq b), x:\overline{[\![b]\!]}\varphi}{x:\neg[\alpha]\varphi} \qquad \frac{x:\overline{[\![b]\!]}\varphi, x:[b]\psi}{x.y:\neg\varphi \mid x.y:\psi}$$

As was pointed out in Section 4, the only causal effect φ of executing b does not imply every necessary effect ψ of executing b. If actions are characterized by their causal effects, necessary effects of an abstract action must also be necessary effects of each refinement. We therefore need a tableau rule that specifies the implied necessary effects of a refinement. We cannot use the rule $\frac{x:\overline{[\![b]\!]}\varphi, x:[b]\psi}{x:\neg[\alpha]\varphi \mid x:[\alpha]\psi}$ because it requires considering every possible action $\alpha \in \mathcal{A}$. However, it is sufficient to consider every linear sequence of atomic actions mentioned in the current leaf of the tableau. These sequences of atomic actions are encoded in the prefixes. $y = \langle a_1\rangle.n_1.\cdots.\langle a_k\rangle.n_k$ in the prefix $x.y$ in the right rule represents a linear sequence of actions $a_1; \ldots; a_k$ that we need to consider.

Note that we do not introduce a rule $\frac{x:\alpha \sqsubseteq b}{x:[\alpha]\psi, x:\overline{[\![b]\!]}\psi}$ because we may not know the 'unique' proposition ψ that completely characterizes the effects of the action b. If we do know the proposition ψ, i.e. $x : \overline{[\![b]\!]}\psi \in \Lambda$, then Proposition 4.7 implies $(\mathcal{I}, s^x) \models [\alpha]\psi$, and the rules, $\frac{x:\overline{[\![b]\!]}\psi}{x:\overline{[\![b]\!]}\psi}$, $\frac{x:[\![b]\!]\psi}{x:[b]\psi, C(x,b) := C(x,b) \cup at(\psi)}$ and $\frac{x:\alpha \sqsubseteq b, x:[b]\psi}{x:[\alpha]\psi}$ enables us to derive $x : [\alpha]\psi$.

Correctness and completeness of tableau rules for characterizing actions by their effects The following lemma addresses the validity of the tableau rules.

LEMMA 5.11
Let Γ be a node of the semantic tableau, and let Γ' and possibly Γ'' be the directly succeeding nodes that are the result of applying one of the three tableau rules listed above.

If Γ is satisfiable, then Γ' or Γ'' is satisfiable.

PROOF. Proposition 4.9 implies the correctness of the left rule, and together with Proposition 3.3, implies the correctness of the right rule. □

For the completeness proof of the reasoning process, we need to be able to construct an interpretation for an open leaf of the tableau. We will adapt the relation R^A of the interpretation $\mathcal{I} = (S, \pi, R^A, C)$ for a saturated open leaf Λ described in Subsection 5.4. If φ describes all necessary

causal effects of the atomic action b, then, based on the assumption that an action is completely characterized by its causal effects, an action α is a refinement of b if φ is a necessary effect of α. Therefore, the states that can be reached by executing α must be among the states that can be causally reached by executing b.

- Let $x : \overline{[\![b]\!]}\varphi \in \Lambda$, and let $E_{s^x,b}$ be the smallest set of states such that for every $\alpha \in \mathcal{A}$, if $(\mathcal{I}, s^x) \models [\alpha]\varphi$, then $\{t \mid (s^x, t) \in R^{\mathcal{A}}(\alpha)\} \subseteq E_{s^x,b}$.
- Then, we define a new interpretation $\mathcal{I}' = (S, \pi, R'^A, C)$ such that for every $b \in A$ and for every $x : \overline{[\![b]\!]}\varphi \in \Lambda$, $(s^x, t) \in R'^A(b)$ iff $t \in E_{s^x,b}$.

The following lemma shows that the adapted interpretation entails Λ.

LEMMA 5.12
Let Λ be an open leaf of a semantic tableau.
Then Λ is satisfiable.

PROOF. We prove the lemma by proving that the adapted interpretation $\mathcal{I}' = (S, \pi, R'^A, C)$ satisfies Λ. The adaptation only influences the case $x : \neg(\alpha \sqsubseteq b) \in \Lambda$, the case $x : \overline{[\![b]\!]}\varphi \in \Lambda$, and the case $x : [b]\psi \in \Lambda$.

- Let $x : \neg(\alpha \sqsubseteq b) \in \Lambda$.
  - Suppose that $x : \overline{[\![b]\!]}\varphi \notin \Lambda$. Then, only the rule $\frac{x:\neg(\alpha \sqsubseteq b)}{x:\langle\alpha\rangle\xi, x:\neg\langle b\rangle\xi}$ has been applied. According to Lemma 5.3, $(\mathcal{I}', s^x) \models \neg(\alpha \sqsubseteq b)$.
  - Suppose that $x : \overline{[\![b]\!]}\varphi \in \Lambda$. Then the rules $\frac{x:\overline{[\![b]\!]}\varphi}{x:[b]\varphi}$ and $\frac{x:\neg(\alpha \sqsubseteq b), x:\overline{[\![b]\!]}\varphi}{x:\neg[\alpha]\varphi}$ must have been applied. Therefore, $x : [b]\varphi \in \Lambda$ and $x : \neg[\alpha]\varphi \in \Lambda$. So, according to the induction hypothesis, $(\mathcal{I}', s^x) \models [b]\varphi$ and $(\mathcal{I}', s^x) \models \neg[\alpha]\varphi$. The latter implies that there is a state $t \in S$ such that $(s^x, t) \in R'^{\mathcal{A}}(\alpha)$ and $(\mathcal{I}', t) \not\models \varphi$. Since $(\mathcal{I}', s^x) \models [b]\varphi$, $(\mathcal{I}', s^x) \models \neg(\alpha \sqsubseteq b)$.
- Let $x : \overline{[\![b]\!]}\varphi \in \Lambda$. The construction of $\mathcal{I}'$ implies that after extending $R^A(b)$ there holds: $(\mathcal{I}', s^x) \models \alpha \sqsubseteq b$ and $(\mathcal{I}', s^x) \models [b]\varphi$. Note that C did not change because of the adaptation of interpretation $\mathcal{I}$. Therefore, since $at(\varphi) \subseteq C(s, b)$, $(\mathcal{I}', s^x) \models [\![b]\!]\varphi$.
  Suppose that $(\mathcal{I}', s^x) \not\models \overline{[\![b]\!]}\varphi$. Then for some $\psi \in \mathcal{L}$, $(\mathcal{I}', s^x) \models [\![b]\!]\psi$ and $\not\models \varphi \rightarrow \psi$. So, $(\mathcal{I}', t) \models \psi$ for every $(s^x, t) \in R'^A(b)$. Since $R^A(b) \subseteq R'^A(b)$ and $(\mathcal{I}, s^x) \models \overline{[\![b]\!]}\varphi$, there is a $(s^x, t) \in R^A(b)$ such that $(\mathcal{I}, t) \not\models \psi$. Contradiction.
  Hence, $(\mathcal{I}', s^x) \models \overline{[\![b]\!]}\varphi$.
- Let $x : [b]\psi \in \Lambda$. It suffices to show that every state in $E_{x,b} \backslash \{t \mid (s^x, t) \in R^A(b)\}$ entails ψ. Let $\alpha \in \mathcal{A}$ be a refinement of b, i.e. $(\mathcal{I}, s^x) \models [\alpha]\varphi$.
  Suppose that $(s^x, t) \in R'^{\mathcal{A}}(\alpha)$. Then there is a prefix $x.y$ such that $y = \langle a_1\rangle.n_1.\cdots.\langle a_k\rangle.n_k$ represents an execution path of α, and $t = s_{n_k} = s^{x.y}$. Since $\{x : \overline{[\![b]\!]}\varphi, x : [b]\psi\} \subseteq \Lambda$ and since $x.y$ is a prefix occurring in Λ, the rule $\frac{x:\overline{[\![b]\!]}\varphi, x:[b]\psi}{x.y:\neg\varphi \mid x.y:\psi}$ must have been applied on the branch containing Λ.
  Suppose that Λ is a leaf of the left sub-branch starting from the rule application. Then, $x.y : \neg\varphi \in \Lambda$. Therefore, according to the induction hypothesis, $(\mathcal{I}, t) \models \neg\varphi$. Since $(s^x, t) \in R'^{\mathcal{A}}(\alpha)$, $(\mathcal{I}, s^x) \not\models [\alpha]\varphi$. Contradiction.
  Hence, Λ is a leaf of the right sub-branch starting from the rule application. Therefore, $x.y : \psi \in \Lambda$.
  Hence, for every state $t \in S$, if $(s^x, t) \in R'^{\mathcal{A}}$, then $(\mathcal{I}, t) \models \psi$. Therefore, $(\mathcal{I}, s^x) \models [b]\psi$. □

5.6 Tableau rules for always and sometimes

We also need tableau rules for a proposition that always holds, and for a proposition that sometimes holds. In the former case, the proposition holds in the current state, and in any state that can be reached by executing an action. In the latter case, the proposition holds in a state that can be reached by executing zero or more actions. The following tableau rules formalize this semantics.

$$\frac{x{:}\neg\Box\varphi}{x{:}\Diamond\neg\varphi} \quad \frac{x{:}\neg\Diamond\varphi}{x{:}\Box\neg\varphi} \quad \frac{x{:}\Box\varphi}{x{:}\varphi} \quad \frac{x{:}\Box\varphi}{x.\langle a\rangle.n{:}\Box\varphi} \quad \frac{x{:}\Diamond\varphi}{x{:}\langle f\rangle\varphi}$$

The prefix $x.\langle a\rangle.n$ with $a \in A$ must already be present in the current leaf. The atomic action f is a *new* action not belonging to A. We can view f as an *abstract* action that represents some composite action $\alpha \in \mathcal{A}$.

Correctness and completeness of tableau rules for always and sometimes The rules for always and sometimes are valid tableau rules.

LEMMA 5.13
Let Γ be a node of the semantic tableau, and let Γ' be the child node that are the result of applying one of the tableau rules for always and sometimes given in Section 2.

If Γ is satisfiable, then Γ' is satisfiable.

PROOF. Note that $\models \Box\theta \leftrightarrow \neg\Diamond\neg\theta$. From this the correctness of the first two rules follows.

Note that $\left(\bigcup_{a\in A} R^A(a)\right)^* = \{(s,s) \mid s \in S\} \cup \left(\bigcup_{a\in A} R^A(a)\right) \circ \left(\bigcup_{a\in A} R^A(a)\right)^*$ So, $\Box\varphi$ is equivalent to $\varphi \wedge \bigwedge_{a\in A}[a]\Box\varphi$. From this result, the correctness of the third and fourth rules follows.

$(\mathcal{I}, s^x) \models \Diamond\varphi$ iff for some $t \in S$, $(s^x, t) \in \left(\bigcup_{a\in A} R^A(a)\right)^*$ and $(\mathcal{I}, t) \models \varphi$. Let f be a new action such that $(s^x, t) \in R^A(f)$. Then, $(\mathcal{I}, s^x) \models \langle f\rangle\varphi$ must hold. From this result, the correctness of the last rule follows.

Hence, If Γ is satisfiable, then Γ' is satisfiable. □

For the completeness proof of the reasoning process, we need to be able to construct an interpretation for an open leaf of the tableau.

LEMMA 5.14
Let Λ be an open leaf of the semantic tableau.

Then we can construct an interpretation $\mathcal{I} = (S, \pi, R^A, C)$ entailing the propositions in Λ.

PROOF. We reuse the interpretation $\mathcal{I} = (S, \pi, R^A, C)$ for a saturated open leaf Λ described in Subsection 5.5. Note that every state in the interpretation $\mathcal{I}$ has a corresponding prefix and that for every prefix $x \neq 0$ there is a prefix y, an action $a \in A$ and a number $n \in \mathbb{N}$ such that $x = y.\langle a\rangle.n$. We can prove that by induction on the length of the prefix x that $(\mathcal{I}, s^x) \models \Box\varphi$ iff $x : \Box\varphi \in \Lambda$.

$(\mathcal{I}, s^x) \models \Diamond\varphi$ iff $x : \Diamond\varphi \in \Lambda$ follows from $(\mathcal{I}, s^x) \models \langle f\rangle\varphi$ iff $x : \langle f\rangle\varphi \in \Lambda$. □

5.7 Correctness and completeness of the tableau method

Based on the lemmas that were proved in the previous subsections, we can prove the following theorem. The theorem guarantees the correctness and completeness of the tableau method for PDLR.

THEOREM 5.15
The semantic tableau method for PDLR is correct

- if the tableau for $\Sigma \cup \{\neg\varphi\}$ closes, then $\Sigma \models \varphi$

and complete

- if $\Sigma \models \varphi$, then the tableau for $\Sigma \cup \{\neg\varphi\}$ closes

PROOF. The correctness follows from the correctness of the original tableau method for PDL [14] and Lemmas 5.2, 5.6, 5.9, 5.11 and 5.13. The lemmas makes it possible to prove that one of the leafs of the tableau is satisfiable if the root of the tableau is satisfiable. The completeness follows from the completeness of the original tableau method for PDL [14] and Lemmas 5.3, 5.7, 5.10, 5.12 and 5.14. The lemmas show that we can construct an interpretation satisfying a saturated open leaf of the tableau. Hence, the leaf is satisfiable. Since a root is a subset of the leaf, the root is also satisfiable. □

6 Relations with other approaches

The BDI model Although BDI logics such as [5, 33] do not consider the refinement of abstract actions, agent programming languages do consider the refinement of abstract actions. For an overview of agent programming languages, see for instance [6, 23]. The semantics of an agent programming language is usually an operational semantics. The above proposed PDLR enables the formalization of a program in an agent programming language in dynamic logic. This enables reasoning about the state changes caused by a program.

We will illustrate the translation of a plan in the language AgentSpeak [32] to a proposition in PDLR. Let $p_1, \ldots, p_m$ be a set of atomic propositions, and let $a_1, \ldots, a_n$ be a set of atomic actions. AgentSpeak distinguishes atomic propositions describing beliefs: $p_1, \ldots, p_m$, achievement goals: $!p_1, \ldots, !p_m$, test goals: $?p_1, \ldots, ?p_m$, atomic actions: $a_1, \ldots, a_n$ and events, which are additions $(+)$ or deletions $(-)$ of beliefs: $+p_1, \ldots, +p_m, -p_1, \ldots, -p_m$, of achievement goals: $+!p_1, \ldots, +!p_m, -!p_1, \ldots, -!p_m$, and of test goals: $+?p_1, \ldots, +?p_m, -?p_1, \ldots, -?p_m$. A plan describes a possible reaction on an event:

$$e : b_1, \ldots, b_k <\!\!- \mathsf{h}_1; \ldots; \mathsf{h}_l,$$

where e is the triggering event, $b_1, \ldots, b_k$ are beliefs, and $\mathsf{h}_1; \ldots; \mathsf{h}_l$ is a plan described by a sequences of goals or actions. Note that goals in the plan trigger new internal events.

The planning part of AgentSpeak starts with an event introducing an achievement goal. This planning part can be reformulated in PDLR. We describe each plan triggered by an achievement goal by a proposition in PDLR. The plan of the form

$$+!g : b_1, \ldots, b_k <\!\!- \mathsf{h}_1; \ldots; \mathsf{h}_l$$

is described by the proposition of the form

$$\Box\left((b_1 \wedge \ldots \wedge b_k) \rightarrow (\mathsf{r}_1; \ldots; \mathsf{r}_l \sqsubseteq \#g)\right),$$

where $\mathsf{r}_i = \mathsf{h}_i$ if h_i is an action, $\mathsf{r}_i = \#g_i$ if $\mathsf{h}_i = !g_i$ and $\mathsf{r}_i = b_i?$ if $\mathsf{h}_i = ?b_i$. Note that $\#p$ is the abstract action denoting the achievement of the proposition p (see the last paragraph of Section 4). Also note that propositions in Σ without a modal operator are interpreted as describing the agent's beliefs. We could also extend PDLR with a modal operator B for beliefs. This would make it possible to describe facts about the world and the agent's beliefs about the world.

EXAMPLE 6.1
The following plan rules implement our 'going to Rome' example in AgentSpeak.

$+!$ 'going to Rome$'$: 'train connection to Rome$'$ <- $!$ 'going to Rome by train$'$
$+!$ 'going to Rome by train$'$: 'bus connection to train station$'$ <-
'take the bus$'$; 'take the train$'$
$+!$ 'going to Rome by train$'$: 'no rain$'$ <- 'walk$'$; 'take the train$'$
$+!$ 'going to Rome$'$: 'car available$'$ <- 'goto Rome by car$'$

We reformulate this in PDLR. Information Σ should contain the propositions:

$\Box$('train connection to Rome$'$ $\rightarrow$ ('goto Rome by train$'$ $\sqsubseteq$ 'goto Rome$'$))
$\Box$('bus connection to train station$'$ $\rightarrow$
('take the bus$'$; 'take the train$'$ $\sqsubseteq$ 'goto Rome by train$'$))
$\Box$('no rain$'$ $\rightarrow$ ('walk$'$; 'take the train$'$ $\sqsubseteq$ 'goto Rome by train$'$))
$\Box$('car available$'$ $\rightarrow$ ('goto Rome by car$'$ $\sqsubseteq$ 'goto Rome$'$))

HTNs The above proposed PDLR enables the formalization of HTNs [13; 36] in dynamic logic. HTNs denote a class of planning problems that differ from classical planning problems. Instead of finding a (partially) ordered set of directly executable actions that brings an agent from an initial state to a goal state, in an HTN, we start with an abstract action, called a *task*, and step by step, refine this task till we end up with a (partially) ordered set of directly executable actions. So, starting from an abstract task such as 'goto Rome' we determine a detailed plan of directly executable actions such as 'take the bus to the train station' followed by 'take the train to Rome', through hierarchical refinements of the original abstract task.

Different definitions of HTNs can be found in the literature. Here, we consider the definition of a simple task networks (STNs) proposed by Ghallab *et al.* [13]. Ghallab *et al.* introduce an STN is a simplified version of an HTN. In an HTN additional constraints can be introduced in a refinement of an abstract task.

An STN consists of a set of tasks $\mathcal{T}$ divided into primitive tasks $\mathcal{T}^p$ and abstract tasks $\mathcal{T}^a$ and a set of methods $\mathcal{M}$. A primitive task can be executed by a corresponding directly executable action a provided that the *precondition* of the action is met. An abstract task can be refined by some *method* $m \in \mathcal{M}$ provided that the *precondition* of the method is met. The method m specifies a (partially) ordered sequence of less abstract tasks. Formally,

$$m \in \mathcal{T}^a \times \mathcal{L} \times 2^{\mathcal{T}} \times 2^{\mathcal{T}\times\mathcal{T}}.$$

So, the method $m = (t, \varphi, \{t_1, \ldots, t_k\}, <)$ specifies the abstract task t to which the method m can be applied, a precondition φ for applying the method m, a refinement of the abstract task t into less abstract tasks $\{t_1, \ldots, t_k\}$, and a partial execution order $< \subseteq \{t_1, \ldots, t_k\} \times \{t_1, \ldots, t_k\}$ on the tasks $t_1, \ldots, t_k$.

We can describe an abstract task t by an abstract action b, a method m with precondition φ by a proposition of the form $\Box(\varphi \rightarrow (\alpha \sqsubseteq b))$ where α describes a partially ordered set of less abstract actions specified by the method m, and a primitive task t and the corresponding action by a directly executable action a with precondition φ and effect ψ: $\Box(\varphi \rightarrow (\langle a\rangle\psi \wedge [a]\psi))$. A directly executable action in PDLR is a non-abstract action. So, an atomic action $a \in A$ is directly executable if every refinement of a is an atomic actions $b \in A$ and a is a refinement of b.

A partially ordered sequence of less abstract tasks corresponds to having choices in the execution order of actions. So, if the execution order of α_1 and α_2 does not matter, then we can specify this by $\alpha_1;\alpha_2 + \alpha_2;\alpha_1$. This translation is not equivalent to having no order on the tasks. Consider and

HTN where $\{b, c\}$ is a refinement of a, $\{b', b''\}$ is a refinement of b with $b' < b''$, and $\{c', c''\}$ is a refinement of c with $c' < c''$. Given an abstract task a, the plan $b'; c'; b''; c''$ can be derived. The translation of an HTN to PDLR does not allow for such plan. If no order on $\{b, c\}$ is specified, we can only translate the refinement of a to $(b; c) + (c; b) \sqsubseteq a$. So, the only possible plans are $b'; b''; c'; c''$ and $c'; c''; b'; b''$.

EXAMPLE 6.2
Consider an HTN planning problem of 'going to Rome'.

- A set of tasks:
  $\mathcal{T} =$ {goto Rome, goto Rome by train, take the train, walk, take the bus}.
- A set of methods: {(goto Rome, train connection to Rome, {goto Rome by train}, $\varnothing$),
  (goto Rome by train, bus connection to train station,
  {take the bus, take the train}, take the bus $<$ take the train),
  (goto Rome by train, no rain, {walk, take the train}, walk $<$ take the train) }

We reformulate this in PDLR. The information Σ should contain the following propositions:

$$\Box(\text{`train connection to Rome'} \rightarrow (\text{`goto Rome by train'} \sqsubseteq \text{`goto Rome'}))$$
$$\Box(\text{`bus connection to train station'} \rightarrow (\text{`take the bus'}; \text{`take the train'} \sqsubseteq \text{`goto Rome by train'}))$$
$$\Box(\text{`no rain'} \rightarrow (\text{`walk'}; \text{`take the train'} \sqsubseteq \text{`goto Rome by train'}))$$

Herzig *et al.* [18] describe rationality postulates that must be satisfied by a well defined HTN planning problem. They assume HTNs that do not have preconditions for applying methods, but do assume that preconditions and effects can be specified for abstract tasks, and therefore for the corresponding abstract actions. The first postulate states that if an abstract action b is executable, every applicable refinement α should realize the effect of executing b. Since the semantics of $\alpha \sqsubseteq b$ guarantees that this always holds in PDLR, the refinement relation has to be considered separately. We reformulate the postulate because of two reasons. Firstly, Herzig *et al.* use planning information as axioms that must be satisfied by every state of an interpretation. We focuses on interpretation-state pairs that satisfy the planning information as well as information about the current state. Secondly, we consider a more general HTN planner that allows for conditional refinements.

> Let Σ specify the precondition and effects of every action / task b by a proposition of the form $\Box(\eta \rightarrow (\langle b\rangle\mu \wedge [b]\mu))$. Moreover, let P specify the refinements of abstract actions by propositions of the form $\Box(\varphi \rightarrow \alpha \sqsubseteq b)$. Finally, let $(\mathcal{I}, s)$ be an interpretation-state pair such that $(\mathcal{I}, s) \models \Sigma$, and let s' be reachable from s, i.e. $(s, s') \in \left(\bigcup_{a\in A} R^A(a)\right)^*$.
> An action b is *soundly refinable* at $(\mathcal{I}, s')$ iff $(\mathcal{I}, s') \not\models \eta$, or for every $\Box(\varphi \rightarrow \alpha \sqsubseteq b) \in P$, if $(\mathcal{I}, s') \models \varphi$, then for every t, if $(s', t) \in R^{\mathcal{A}}(\alpha)$, then $(\mathcal{I}, t) \models \mu$.
> We say that an action b is soundly refinable iff for every $(\mathcal{I}, s)$ such that $(\mathcal{I}, s) \models \Sigma$, b is soundly refinable at every $(\mathcal{I}, s')$ such that s' is reachable from s.

Herzig *et al.*'s theorem has to be adapted too. We can now prove the following theorem.

THEOREM 6.3
Let Σ specify the precondition and effects of every action/task b by a proposition of the form $\Box(\eta \rightarrow (\langle b\rangle\mu \wedge [b]\mu))$. Moreover, let P specify the refinements of abstract actions by propositions of the form $\Box(\varphi \rightarrow \alpha \sqsubseteq b)$.

An action b is *soundly refinable* iff for every $\Box(\eta \rightarrow (\langle b\rangle\mu \wedge [b]\mu)) \in \Sigma$ and for every $\Box(\varphi \rightarrow \alpha \sqsubseteq b) \in P$, $\Sigma \models \Box((\eta \wedge \varphi) \rightarrow [\alpha]\mu)$.

PROOF. Let $(\mathcal{I}, s)$ interpretation-state pair such that $(\mathcal{I}, s) \models \Sigma$, and let s' be reachable from s, i.e. $(s, s') \in \left(\bigcup_{a\in A} R^A(a)\right)^*$. Clearly, $(\mathcal{I}, s') \models \Sigma$.

Let $\Box(\eta \rightarrow (\langle b\rangle\mu \wedge [b]\mu)) \in \Sigma$.

- If $(\mathcal{I}, s') \not\models \eta$, then $(\mathcal{I}, s') \models \Box((\eta \wedge \varphi) \rightarrow [\alpha]\mu)$.
- If $(\mathcal{I}, s') \models \eta$, then $(\mathcal{I}, s') \models [b]\mu$. Since b is soundly refinable, if $(\mathcal{I}, s') \models \varphi$, then $(\mathcal{I}, s') \models [\alpha]\mu$. Hence, $(\mathcal{I}, s') \models \Box((\eta \wedge \varphi) \rightarrow [\alpha]\mu)$.

Hence, $\Sigma \models \Box((\eta \wedge \varphi) \rightarrow [\alpha]\mu)$. □

Herzig *et al.*'s second postulate states that every abstract action must have a refinement. This postulate has also been reformulated and the theorem that is derived from it has been adapted.

> Let Σ specify the precondition and effects of every action/task b by a proposition of the form $\Box(\eta \rightarrow (\langle b\rangle\mu \wedge [b]\mu))$. Moreover, let P specify the refinements of abstract actions by propositions of the form $\Box(\varphi \rightarrow \alpha \sqsubseteq b)$. Finally, let $(\mathcal{I}, s)$ be an interpretation-state pair such that $(\mathcal{I}, s) \models \Sigma$ and let s' be reachable from s, i.e. $(s, s') \in \left(\bigcup_{a\in A} R^A(a)\right)^*$.
> An action b is *completely refinable* at $(\mathcal{I}, s')$ iff $(\mathcal{I}, s') \not\models \eta$, or for some $\Box(\varphi \rightarrow \alpha \sqsubseteq b) \in P$, $(\mathcal{I}, s') \models \varphi$ and there exists a t such that $(s', t) \in R^{\mathcal{A}}(\alpha)$.
> We say that an action b is completely refinable iff for every $(\mathcal{I}, s)$ such that $(\mathcal{I}, s) \models \Sigma$, b is completely refinable at every $(\mathcal{I}, s')$ such that s' is reachable from s.

THEOREM 6.4
Let Σ specify the precondition and effects of every action / task b by a proposition of the form $\Box(\eta \rightarrow (\langle b\rangle\mu \wedge [b]\mu))$. Moreover, let P specify the refinements of abstract actions by propositions of the form $\Box(\varphi \rightarrow \alpha \sqsubseteq b)$.

An action b is *completely refinable* iff for every $\Box(\eta \rightarrow (\langle b\rangle\mu \wedge [b]\mu)) \in \Sigma$, $\Sigma \models \Box(\eta \rightarrow \langle +_{\Box(\varphi\rightarrow\alpha\sqsubseteq b)\in P}(\varphi?;\alpha)\rangle\top)$.[16]

PROOF. Let $(\mathcal{I}, s)$ interpretation-state pair such that $(\mathcal{I}, s) \models \Sigma$, and let s' be reachable from s, i.e. $(s, s') \in \left(\bigcup_{a\in A} R^A(a)\right)^*$. Clearly, $(\mathcal{I}, s') \models \Sigma$.

Let $\Box(\eta \rightarrow (\langle b\rangle\mu \wedge [b]\mu)) \in \Sigma$.

- If $(\mathcal{I}, s') \not\models \eta$, then $\Sigma \models \Box(\eta \rightarrow \langle +_{\Box(\varphi\rightarrow\alpha\sqsubseteq b)\in P}(\varphi?;\alpha)\rangle\top)$.
- If $(\mathcal{I}, s') \models \eta$, then $(\mathcal{I}, s') \models (\langle b\rangle\mu \wedge [b]\mu)$. Since b is completely refinable, there exists a $\Box(\varphi \rightarrow \alpha \sqsubseteq b) \in P$ such that $(\mathcal{I}, s') \models \varphi$ and $(\mathcal{I}, s') \models \langle\alpha\rangle\top$. Therefore, there exists a $\Box(\varphi \rightarrow \alpha \sqsubseteq b) \in P$ such that $(\mathcal{I}, s') \models \langle\varphi?;\alpha\rangle\top$. Hence, $(\mathcal{I}, s') \models \langle +_{\Box(\varphi\rightarrow\alpha\sqsubseteq b)\in P}(\varphi?;\alpha)\rangle\top$, and $\Sigma \models \Box(\eta \rightarrow \langle +_{\Box(\varphi\rightarrow\alpha\sqsubseteq b)\in P}(\varphi?;\alpha)\rangle\top)$.

Hence, $\Sigma \models \Box(\eta \rightarrow \langle +_{\Box(\varphi\rightarrow\alpha\sqsubseteq b)\in P}(\varphi?;\alpha)\rangle\top)$. □

Herzig *et al.* claim that PDL extended with a refinement relation is undecidable because of the correspondence with grammar logics. As mentioned above, no proof is given for this claim and it is unclear how the undecidability results for grammar logics imply the undecidability of PDLR. Although the claim might be correct, the restriction that the right-hand side of the refinement relation to a single abstract action guarantees the decidability of PDLR. The methods for refining abstract

[16] $+_{\Box(\varphi\rightarrow\alpha\sqsubseteq b)\in P}(\varphi?;\alpha)$ denotes $(\varphi_1?;\alpha_1) + \ldots + (\varphi_k?;\alpha_k)$ where $P = \{\Box(\varphi_1 \rightarrow \alpha_1 \sqsubseteq b), \ldots, \Box(\varphi_k \rightarrow \alpha_k \sqsubseteq b)\}$.

tasks in an HTN's use the same restriction. Nevertheless, the plan existence problem for HTNs is undecidable [12]. The undecidability is the result of methods that do not specify an order for the refinement. To give an illustration, let $\{b, c\}$ be a refinement of a, $\{b', b''\}$ be a refinement of b with $b' < b''$, and $\{c', c''\}$ be a refinement of c with $c' < c''$. Then we can derive a plan $b'; c'; b''; c''$. Such a plan is used in the undecidability proof of HTNs. As was already pointed out above, the translation of an HTN to PDLR does not allow for such plan. Since the plan $b'; c'; b''; c''$ is not possible in PDLR, the undecidability proof given in [12] cannot be constructed for PDLR.

The reformulation of an HTN in terms of PDLR can be viewed as a way to provide a semantics for HTNs [18]. The soundness of the practical implementation of an HTN planner can be verified w.r.t. the PDLR formulation. As we have seen above, there is a difference between unordered abstract tasks and the corresponding abstract actions. If no execution ordering is specified for a set of abstract tasks, then we must express all possible execution orderings of the corresponding set of abstract actions. Because of this difference, the plan existence problem [12], which is undecidable for HTNs, is decidable for PDLR. In PDLR, the plan existence problem is equivalent to abstract actions being soundly and completely refinable; see Theorems 6.3 and 6.4. Abstract actions being soundly and completely refinable are decidable properties because PDL proofs are decidable.

Temporal constraints are often important in planning problems. For instance, being in Rome next Friday afternoon, a flight of 3 hours, a train journey of one and a half day, and the schedule of the airplanes and the trains. The temporal constraints can for instance be handled by combining HTNs with Simple Temporal Networks [31]. A proper handling of temporal constraints requires first-order dynamic logic.

7 Conclusion

This paper investigated PDLR which extends PDL with a refinement relation that refines an abstract action into a more specific (composite) action. We conclude that such an extension of PDL is possible. Moreover, we conclude that it is also possible to derive the refinement relation if we describe all the necessary causal effects of an abstract action execution. To describe all necessary causal effects of an action execution, an operator that modifies a modal necessity operator was introduced. Applying this new operator to the knowledge operator in epistemic logic, introduces *only-knowing* in epistemic logic. Finally, we conclude that a proof system based on prefixed-tableaux can be defined. Correctness and completeness of the semantic tableau method is proved.

This research offers several possibilities for further research. Firstly, it is interesting to investigate whether the worst-case computational complexity is affected by the proposed extensions of PDL. Secondly, the computational complexity in practical application can be investigated. Thirdly, the extension to first-order dynamic logic can be investigated. Finally, the causal effect of composite actions, which is related to the handling of the frame problem, can be addressed.

Responsible AI is an important topic nowadays and the possibility of deriving new refinements of abstract actions raises ethical issues. Given the abstract action of 'getting a new car' we should consider the refinement of 'stealing a car'. This is of course not only an issue for PDL extended with the refinement of abstract actions, but for any planning system. The integration of ethical norms in systems that can derive new plans/refinements is also an important topic for further research.

Acknowledgment

I thank the reviewers for their valuable comments, which helped me to improve the paper.

References

[1] P. Blackburn. Representation, reasoning, and relational structures: A hybrid logic manifesto. *Logic Journal of the IGPL*, **8**, 339–365, 2000.

[2] P. Blackburn, M. de Rijke and Y. Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

[3] M. E. Bratman. *Intention, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA, 1987.

[4] M. A. Castilho, O. Gasquet and A. Herzig. Formalizing action and change in modal logic I: The frame problem. *Journal of Logic and Computation*, **9**, 701–735, 1999.

[5] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, **42**, 213–261, 1990.

[6] M. Dastani. A survey of multi-agent programming languages and frameworks. In *Agent-Oriented Software Engineering—Reflections on Architectures, Methodologies, Languages, and Frameworks*, O. Shehory and A. Sturm, eds, pp. 213–233. Springer, 2014.

[7] L. F. del Cerro and M. Penttonen. Grammar logics. *Logique et Analyse*, **31**, 123–134, 1988.

[8] R. Demolombe, A. Herzig and I. Varzinczak. Regression in modal logic. *Journal of Applied Non-Classical Logics*, **13**, 165–185, 2003.

[9] S. Demri. The complexity of regularity in grammar logics and related modal logics. *Journal of Logic and Computation*, **11**, 933–960 12, 2001.

[10] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, **2**, 189–208, 1971.

[11] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, **18**, 194–211, 1979.

[12] T. Geier and P. Bercher. On the decidability of HTN planning with task insertion. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence—Volume Three, IJCAI'11*, pp. 1955–1961. AAAI Press, 2011.

[13] M. Ghallab, D. Nau and P. Traverso. *Automated Planning*. Elsevier, 2004.

[14] G. De Giacomo and F. Massacci. Tableaux and algorithms for propositional dynamic logic with converse. In *Proceedings of the 13th International Conference on Automated Deduction, CADE-13*, pp. 613–627. Springer, 1996.

[15] G. De Giacomo and F. Massacci. Combining deduction and model checking into tableaux and algorithms for converse-PDL. *Information and Computation*, **162**, 117–137, 2002.

[16] R. Goré and F. Widmann. Optimal and cut-free tableaux for propositional dynamic logic with converse. In *Automated Reasoning, 5th International Joint Conference, IJCAR 2010*, J. Giesl and R. Hähnle, eds, Edinburgh, UK, 16–19 July 2010. Vol. 6173 of Lecture Notes in Computer Science, pp. 225–239. Springer, 2010.

[17] A. Herzig, E. Lorini, L. Perrussel and Z. Xiao. BDI logics for BDI architectures: Old problems, new perspectives. *Künstliche Intelligenz*, **31**, 73–83, 2017.

[18] A. Herzig, L. Perrussel and Z. Xiao. On hierarchical task networks. In *Logics in Artificial Intelligence: 15th European Conference, JELIA 2016*, L. Michael and A. C. Kakas, eds, Larnaca, Cyprus, 9–11 November 2016. Vol. 10021 of Lecture Notes in Computer Science, pp. 551–557. Springer, 2016.

[19] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, **9**, 385–410, 1999.

[20] U. Hustadt and R. A. Schmidt. A comparison of solvers for propositional dynamic logic. In *Proceedings of the 2nd Workshop on Practical Aspects of Automated Reasoning, PAAR-2010*,

R. A. Schmidt, S. Schulz and B. Konev, eds, Edinburgh, Scotland, UK, 14 July 2010. Vol. 9 of EPiC Series in Computing, pp. 63–73. EasyChair, 2010.

[21] G. Lakemeyer. The situation calculus: A case for modal logic. *Journal of Logic, Language and Information*, **19**, 431–450, 2010.

[22] H. J. Levesque. All I know: A study in autoepistemic logic. *Artificial Intelligence*, **42**, 263–309, 1990.

[23] V. Mascardi, D. Demergasso and D. Ancona. Languages for programming BDI-style agents: An overview. In *WOA 2005: Dagli Oggetti Agli Agenti. 6th AI*IA/TABOO Joint Workshop "From Objects to Agents": Simulation and Formal Analysis of Complex Systems*, F. Corradini, F. De Paoli, E. Merelli and A. Omicini, eds, Camerino, MC, Italy, 14–16 November 2005, pp. 9–15. Pitagora Editrice, Bologna, 2005.

[24] F. Massacci. Single step tableaux for modal logics: Computational properties, complexity and methodology. *Journal of Automated Reasoning*, **24**, 319–364, 2000.

[25] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, pp. 463–502. Edinburgh University Press, 1969.

[26] J.-J. C. Meyer, J. M. Broersen and A. Herzig. BDI logics. In *Handbook of Logics of Knowledge and Belief*, Chapter 10, H. van Ditmarsch, J. Y. Halpern, W. van der Hoek and B. Kooi, eds, pp. 453–498. College Publications, 2015.

[27] R. C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, **25**, 75–94, 1985.

[28] L. A. Nguyen and A. Szałas. An optimal tableau decision procedure for Converse-PDL. In *Proceedings of KSE'2009*, N.-T. Nguyen, T.-D. Bui, E. Szczerbicki and N.-B. Nguyen, eds, pp. 207–214. IEEE Computer Society, 2009.

[29] E. P. D. Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pp. 324–332. Morgan Kaufmann Publishers, 1989.

[30] V. Pratt. Semantical considerations on Floyd–Hoare logic. In *Proceedings of the 17th IEEE Symposium on Foundations of Computer Science*, pp. 109–121. IEEE Computer Society, 1976.

[31] C. Qi, D. Wang, H. Muñoz-Avila, P. Zhao and H. Wang. Hierarchical task network planning with resources and temporal constraints. *Knowledge-Based Systems*, **133**, 17–32, 2017.

[32] A. S. Rao. Agentspeak(L): BDI agents speak out in a logical computable language. In *MAAMAW*. Vol. 1038 of Lecture Notes in Computer Science, pp. 42–55. Springer, 1996.

[33] A. S. Rao and M. P. Georgeff. Modeling Rational Agents within a BDI-Architecture. In *Proceedings of the Second International Conference op Principles of Knowledge Representation and Reasoning (KR),* pp. 473–484. Morgan Kaufmann Publishers, 1991.

[34] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, 2001.

[35] N. Roos. Semantics of behavior. In *European Conference on Artificial Intelligence (ECAI)*, pp. 771–776. IOS Press, 2014.

[36] E. D. Sacerdoti. *A Structure for Plans and Behavior*. Elsevier-North Holland, 1977.

[37] S. Sardina, L. de Silva and P. Lin. Hierarchical planning in BDI agent programming languages: A formal approach. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), AAMAS '06*, pp. 1001–1008. Association for Computing Machinery, New York, NY, USA, 2006.

[38] D. Zhang and N. Foo. EPDL: A logic for causal reasoning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI01)*, pp. 131–138. Morgan Kaufmann Publishers, 2001.

[39] D. Zhang and N. Y. Foo. Frame problem in dynamic logic. *Journal of Applied Non-Classical Logics*, **15**, 215–239, 2005.

[40] Y. Zhou and Y. Zhang. Modeling abstract behavior: A dynamic logic approach. In *AI 2009: Advances in Artificial Intelligence*, A. Nicholson and X. Li, eds. Vol. 5866 of Lecture Notes in Computer Science, pp. 538–546. Springer, 2009.

Received 1 May 2020