Tableau-Kalkül für PDL und Interpolation

Tableau Calculus for PDL and Interpolation

Manfred Borzechowski

translated by Malvin Gattinger

Written at
Freie Universität Berlin, 1988

Translated at
Rijksuniversiteit Groningen, 2020

Last update of this file: January 17, 2025

Contents

Preface to the Translation

The question whether Propositional Dynamic Logic has Craig Interpolation has gotten much attention. There have been at least three proof attempts:

- Daniel Leviant in 1981 (`https://doi.org/10.1007/3-540-10699-5_111`)
- Manfred Borzechowski in 1988 (`http://www.borzechowski.de/TableauKalkülFürPDLUndInterpolation.pdf`)
- Tomasz Kowalski in 2002 (`https://doi.org/10.2178/jsl/1190150141`)

The proof by Kowalski was officially retracted (in 2004), but the status of the other two proofs is unclear. One might even say that it is an open question whether it is an open question whether PDL has CI.

In his book *Tools and Techniques in Modal Logic*, Marcus Kracht devotes a chapter to "The Unanswered Question" and mentions both proofs by Leivant and Borzechowski. Kracht points out a problem in the proof by Leivant which I discussed in `https://w4eg.de/malvin/illc/pdl.pdf`. I believe that the finitary rule criticised by Kracht is *not* a problem, but there are other steps in the proof which could not be clarified so far, including the "$\neg\neg$ trick".

Interestingly, Kracht does not write anything further about Borzechowskis proof. Borzechowski wrote his diploma thesis in German and never published it. My impression is that very few people actually read it. By translating it from German to English (and from scanned Atari 800XL to LaTeX) I hope to change this. Maybe the answer has been waiting for us since 1988.

A few notes on the translation follow. I try to stay as close to the original as possible without violating English grammar. Some concepts I translate more freely. For example, "Endlichkeitssatz für die Folgerungsrelation" is simply the compactness theorem. Selected vocabulary can be found at the end of the document. For example, I decided to translate "belastet" as 'loaded'.

The margins show the original page numbers and additions are grey. When finding any errors the reader should of course first assume that I introduced them and check the original. Please send corrections to malvin@w4eg.eu.[1]

Malvin Gattinger

February 2020
Groningen

[1] I thank Gerard Renardel and Jan Rooduijn for pointing out typos.

↓ page 3

0 Introduction

During the last twenty years (i.e. 1968–1988) many logics and calculi have been developed and studied in which properties of *programs* can be formalised and formally proven. These logics are not just interesting from a theoretical point of view: they are also a necessary tool to solve problems in program verification and automated program synthesis.

One of these logics is *dynamic logic* (*DL*) from 1976 (PRATT [7]). It is a generalisation of first order modal logic. Its propositional fragment, the *propositional dynamic logic* (*PDL*) was introduced in 1977 (FISCHER/LADNER [4]) and proven to be decidable. Since then not only Theoretical Computer Science, but also Mathematics has studied DL, PDL and their extensions. For example there is a big interest in which properties of basic modal logic are shared by these logics and to what degree they can be proven by established modal logic methods. An exhaustive review of works on this topic can be found in the bibliography of HAREL [5] and in MÜLLER,RAUTENBERG [6].

In the first part of this thesis a tableau-calculus for PDL is developed which builds upon the decision procedure suggested by PRATT [8]. In the second part this calculus is used to show, for the first time, that PDL has the interpolation property.

↓ page 4

1 Tableau for PDL

1.1 The Syntax of PDL

In the language of propositional dynamic logic there are two kinds of expressions: formulas and programs. Let $\mathcal{F}_0$ and $\mathcal{P}_0$ be two countable sets, whose elements we call *propositional variables* and *atomic programs*, respectively. With $p, q, r, \ldots$ we denote the propositional variables and with $A, B, C, \ldots$ we denote the atomic programs. From those sets and the symbols 0, $\neg$, $\wedge$, ;, $\cup$, $*$, ?, [,], (and) we inductively define formulas and programs.

Definition 1. *Every propositional variable and* 0 *is a formula, and every atomic program is a program. In addition, whenever* P *and* Q *are arbitrary formulas and* a *and* b *are arbitrary programs, then the strings*

- $\neg P$, $(P \wedge Q)$ *and* $[a]P$ *are also formulas,*
- $(a;b)$, $(a \cup b)$, a^* *and* $P?$ *are also programs.*

Therefore for example $([(A; B^*)]p \wedge q)$ is a formula. In general we omit the outermost pair of parentheses of formulas and programs; this formula would thus be written shorter as $[A; B^*]p \wedge q$. With $P, Q, R, \ldots$ we denote arbitrary formulas and with $a, b, c, \ldots$ arbitrary programs. Further we write $P_1 \wedge \ldots \wedge P_n$ for $(P_1 \wedge \ldots (P_{n-1} \wedge P_n) \ldots)$ and analogously $a_1; \ldots; a_n$ for $(a_1; \ldots (a_{n-1}; a_n) \ldots)$. If all a_i are the same a, then we write a^n for this. Moreover, let $a^1 := a$ and $a^0 := \neg 0?$. We also use the common abbreviations 1, $P \vee Q$, $P \to Q$ and $P \leftrightarrow Q$ for $\neg 0$, $\neg(\neg P \wedge \neg Q)$, $\neg(P \wedge \neg Q)$ and $(P \to Q) \wedge (Q \to P)$.

↓ page 5

With $\mathcal{F}$ we denote the set of all formulas and with $\mathcal{P}$ the set of all programs. With $X, Y, Z, \ldots$ we always denote finite sets of formulas. We write $X; Y$ for $X \cup Y$ and $X; P$ for $X \cup \{P\}$.

We call a string of the form $[a]$ a *program operator*. As we will see later, in the semantics each program operator $[A]$ with $A \in \mathcal{P}_0$ will play a role similar to the modal operator in the modal logic K. Often in the literature the formulas of PDL are built using program operators of the form $\langle a \rangle$. These are *dual* to the $[a]$ operators; one can either define $[a]P := \neg\langle a \rangle \neg P$ or instead $\langle a \rangle P := \neg[a]\neg P$.

Programs of the form $P?$ are called *tests*. Because of tests both formulas and programs can be built using formulas *and* programs. For example, the formula $[[B]p?; A^*]q$ contains the program $[B]p?$ which in turn contains the formula $[B]P$. This nested construction of formulas and programs means that we often have to use the following proof method.

Simultaneous induction over formula and program construction
Suppose the property E holds for all $p \in \mathcal{F}_0$ and for 0 and suppose the property E' holds for all $A \in \mathcal{P}_0$. Moreover, suppose the following condition holds:

> If E holds for P and Q and E' holds for a and b, then E also holds for $\neg P$, $(P \wedge Q)$, and $[a]P$, and E' also holds for $(a; b)$, $(a \cup b)$, a^* and $P?$.

Then E holds for all $P \in \mathcal{F}$ and E' holds for all $a \in \mathcal{P}$.

↓ page 6

1.2 The Semantics of PDL

We now present the semantics of PDL. It is a generalisation of the relational semantics of modal logic. We note that there is also an algebraic semantics for PDL. The reader interested in *dynamic algebras* is referred to works by PRATT, KOZEN and REITERMAN/TRNKOVA cited in [6].

Definition 2. *A* Kripke model *is a structure of the form (g, π, ν). Here g is a non-empty set whose elements we call* states *and denote with $S, T, U, \ldots$. Furthermore π is a function, which maps each atomic program A to a* reachability relation $\pi(A) \subseteq g \times g$. *Finally, ν is a function which maps each propositional variable p to a set $\nu(p) \subseteq g$, its* extension.

Let us now fix $\mu = (g, \pi, \nu)$ to be a Kripke model. We show how formulas and programs can be interpreted over μ.

Definition 3. *We define in parallel the* acceptance relation $\Vdash^\mu \subseteq g \times (\mathcal{F} \setminus \{0\})$ *and the inductive continuation of π to $\mathcal{P}$. Here we write $S \Vdash P$ for $(S, P) \in \Vdash^\mu$.*

$$
\begin{array}{ll}
S \Vdash p & :\Longleftrightarrow\ S \in \nu(p) \\
S \Vdash P \wedge Q & :\Longleftrightarrow\ S \Vdash P \text{ and } S \Vdash Q \\
S \Vdash \neg P & :\Longleftrightarrow\ \text{not } S \Vdash P \\
S \Vdash [a]P & :\Longleftrightarrow\ T \Vdash P \text{ for all } T \text{ with } (S,T) \in \pi(a)
\end{array}
$$

$$
\begin{array}{ll}
\pi(a \cup b) & := \pi(a) \cup \pi(b) \\
\pi(a;b) & := \pi(a) \circ \pi(b) \\
\pi(a^*) & := \bigcup\{\pi(a^i) \mid i \in \omega\} \\
\pi(P?) & := \{(S,S) \mid S \Vdash P\}
\end{array}
$$

We note that $S \nVdash 0$ and $(S,S) \in \pi(a^*)$ hold for all $S \in g$ and $a \in \mathcal{P}$. For $(S,T) \in \pi(a)$ we often also write $S \xrightarrow{a} T$.

↓ page 7

The reachability relation of non-atomic programs therefore has the following properties:

$$
\begin{array}{lcl}
S \xrightarrow{a;b} T & \Longleftrightarrow & S \xrightarrow{a} U \text{ and } U \xrightarrow{b} T \text{ for some } U \in g \\
S \xrightarrow{a \cup b} T & \Longleftrightarrow & S \xrightarrow{a} T \text{ or } S \xrightarrow{b} T \\
S \xrightarrow{a^*} T & \Longleftrightarrow & S \xrightarrow{a^n} T \text{ for some } n \in \omega \\
S \xrightarrow{P?} T & \Longleftrightarrow & S = T \text{ and } S \Vdash P
\end{array}
$$

A Kripke model can be understood as a model of a computer. One should imagine g as the set of all states in which the computer can be, defined for example as the set of all possible contents of all memory. Each $p \in \mathcal{F}_0$ represents a certain property which holds exactly for the states in $\nu(p)$. Moreover, each atomic program $A \in \mathcal{P}_0$ corresponds to a certain command which can be executed by the CPU. We then have $(S,T) \in \pi(A)$ exactly if T is a possible outcome of executing A starting in state S. And we have $S \Vdash [A]p$ if and only if $T \Vdash p$ holds for all possible resulting states of this execution.

In this way of interpreting Kripke models certain programs can be interpreted as common control structures of imperative programming languages such as PASCAL or ALGOL. This is the case for example for the programs

$$\begin{array}{lcl} (P?;a) \cup (\neg P?;b) & \equiv & \text{IF } P \text{ THEN } a \text{ ELSE } b, \\ a;(\neg P?;a)^*;P? & \equiv & \text{REPEAT } a \text{ UNTIL } P, \\ (P?;a)^*;\neg P? & \equiv & \text{WHILE } P \text{ DO } a. \end{array}$$

Let us convince ourselves in one of these cases. We have

$$\begin{array}{ll} & S \overset{a;(\neg P?;a)^*;P?}{\to} T \\ \iff & S \overset{a;(\neg P?;a)^n}{\to} T \text{ for some } n \text{ and } T \Vdash P \\ \iff & \text{there are } S_i \text{ for } i \leq n \text{ with} \\ & S \overset{a}{\to} S_0 \overset{a}{\to} \ldots \overset{a}{\to} S_n = T \text{ and } S_i \Vdash \neg P \iff i < n \end{array}$$

and this is the case exactly when T is a possible outcome of executing "REPEAT a UNTIL P" starting in state S. This includes a correction: there must be at least one $\overset{a}{\to}$ step. In the original text the prefix $a;$ is missing.

↓ page 8

Definition 4. *We say that a formula P is* satisfiable *if there is a Kripke model (g, π, ν) (corrected from (g, ν, π)) and an $S \in g$ such that $S \Vdash P$ (which we read as "S accepts P" or "P holds in S"). Furthermore, P is called* valid *if $\neg P$ is not satisfiable.*

Example 1. *The following formulas are valid for all P, Q, a, b:*

$$\begin{array}{ll} [a](P \to Q) \to ([a]P \to [a]Q) & [a \cup b]P \leftrightarrow [a]P \wedge [b]P \\ [a^*]P \leftrightarrow (P \wedge [a][a^*]P) & P \wedge [a^*](P \to [a]P) \to [a^*]P \end{array}$$

The bottom right formula is the well-known induction axiom.

Remark 1. *We quickly mention some connections between PDL and modal logic. Let $\mathcal{F}(a)$ denote the set of all formulas in $\mathcal{F}$ which only use the program operator $[a]$ and let $\mathcal{F}^1(a)$ denote the set of all valid formulas in $\mathcal{F}(a)$. Each set $\mathcal{F}^1(a)$ is essentially a normal modal logic (RAUTENBERG [11]); one merely has to identify the modal operator with the program operator $[a]$. Some well-known modal logics can be obtained by choosing a suitable a. For example we have:*

$$\mathsf{K} = \mathcal{F}^1(A) \quad \mathsf{S4} = \mathcal{F}^1(A^*) \quad \mathsf{M} = \mathcal{F}^1(A \cup 1?)$$

This is because the possible reachability relations $\pi(A \cup 1?)$ and $\pi(A^*)$ of all Kripke models over a set g are exactly all reflexive and transitive relations over g, respectively.

The logic CPDL is an extension of PDL obtained by adding the functor $(\cdot)^-$ *(HAREL [5]), interpreted by* $S \xrightarrow{a^-} T \iff T \xrightarrow{a} S$*. With this we have:*

$$\mathsf{B} = \mathcal{F}^1(A \cup 1? \cup A^-) \quad \mathsf{S5} = \mathcal{F}^1((A \cup A^-)^*)$$

The possible reachability relations $\pi(A \cup 1 \cup A^-)$ and $\pi((A \cup A^-)^*)$ of all Kripke models over a set g are exactly all the symmetric-and-reflexive relations and equivalence relations over g, respectively.

↓ page 9

Definition 5. *Let* X *be a set of formulas and* P *be a formula. If for every Kripke model* (g, ν, π) *and every state* $S \in g$ $S \Vdash X$ *implies* $S \Vdash P$*, then we write* $X \vDash P$ *(which we read "from* S *follows* P*").*

We note that in PDL there is no compactness theorem. For example, let $X = \{[A^i]p \mid i \in \omega\}$ and $P = [A^*]p$. Then we have $X \vDash P$. However, for all proper subsets of $Y \subsetneq X$ we have $Y \nvDash P$. To see this, take any $[A^n]p \notin Y$. In every Kripke model $\mu = (g, \nu, \pi)$ with $g = \omega$, $\nu(p) = \omega \setminus \{n\}$ and $\pi(A) = \{(i, i+1) \mid i \in \omega\}$ we have $0 \Vdash^\mu Y$ and $0 \Vdash^\mu \neg[A^n]p$. In particular there is not finite set $X_0 \subseteq X$ such that $X_0 \vDash P$.

↓ page 10

1.3 Decision Procedure for PDL

One of the earliest discoveries about PDL is the *small model property* shown in FISCHER, LADNER [4].

Theorem 1. *If* P *is satisfiable, then* P *is already satisfiable in a Kripke model* (g, ν, π) *such that* $|g| \leq 2^{l(P)}$*.*

Here $l(P)$ denotes the length of the string P. The proof of the theorem uses the *filtration* technique well-known from modal logic. We sketch the proof idea here.

Proof. Let P be satisfiable and $l(P) = n$. Let $c(P)$ denote the Fischer-Ladner closure, i.e. the smallest set that contains P, that is closed under taking subformulas and fulfilling the following conditions:

$$\begin{array}{lcl} [a;b]R \in c(P) & \Rightarrow & [a][b]R \in c(P) \\ [a \cup b]R \in c(P) & \Rightarrow & [a]R, [b]R \in c(P) \\ [a^*]R \in c(P) & \Rightarrow & [a][a^*]R \in c(P) \\ [Q?]R \in c(P) & \Rightarrow & Q \in c(P) \end{array}$$

One shows by induction that we always have $|c(P)| \leq l(P)$. Let now $\mu = (g, \nu, \pi)$ be a Kripke model such that there is a $S_P \in g$ with $S_P \Vdash P$. For

$S, T \in g$ let

$$\begin{aligned} S \equiv_P T \quad &:\Longleftrightarrow \quad (\ S \Vdash R \iff T \vDash R \text{ for all } R \in c(P)\ ) \\ |S|_P \qquad &:= \qquad \{T \mid T \equiv_P S\}. \end{aligned}$$

We now define a Kripke model (g_P, ν_P, π_P) with $g_P := \{|S|_P \mid S \in g\}$. Because of $|c(P)| \leq n$ we have $|g_P| \leq 2^n$. With the definitions

$$\begin{aligned} |S|_P \in \nu_P(p) \qquad &:\Longleftrightarrow \quad \text{there exists } S' \in |S|_P \text{ such that } S' \in \nu(p) \\ (|S|_P, |T|_P) \in \pi_P(A) \quad &:\Longleftrightarrow \quad \text{there exist } S' \in |S|_P \text{ and } T' \in |T|_P \\ &\qquad\qquad\quad \text{such that } (S', T') \in \pi(A) \end{aligned}$$

the model $\mu' = (g_P, \nu_P, \pi_P)$ has the following property:

$$\text{For all } R \in c(P) \text{ and } S \in g, \text{ we have } S \Vdash^{\mu} R \iff |S|_P \Vdash^{\mu'} R$$

Hence in particular we have $|S_P|_P \Vdash^{\mu'} P$. □

From this theorem we can directly derive a decision procedure to check the satisfiability of a formula P:

↓ page 11

> For all Kripke models (g, ν, π) with (for example) $g = \{1, \ldots, K\}$, check whether they have a state $S \in G$ such that $S \Vdash P$.

However, this procedure is completely infeasible. To be sure, the time needed to check whether there exists an $S \in g$ with $S \Vdash P$ is only a polynomial in $|g| + l(P)$ (FISCHER, LADNER [4]). But the number of Kripke models to check is in the order $2^{4^{l(P)}}$ (PRATT [8]), which excludes any practical use of the procedure for other than trivial cases. This motivates the search for better decision procedures, some of which we will present in the following.

Already PRATT [8] uses a satisfiability decision procedure of the same kind as we will do here; namely a Tableau-Calculus. The satisfiability of P is first reduced to the existence of certain structures; In PRATT, those are *filtered tableau structures*, whereas we will use a direct generalisation of *model graphs* from RAUTENBERG [12]. In both cases one shows that such a structure exists if and only if it is not possible to construct certain tableaus. The tableaus we define are also a direct generalisation of the modal logic tableaus defined in RAUTENBERG [12]. As there, also here they allow us to show an interpolation theorem.

Per the theorem above any satisfiable formula P already holds at a state in a Kripke model (g, ν, π) in which every state can be identified with the set $\{R \in c(P) \mid S \Vdash R\}$. The methods used in PRATT [9] and HAREL [5] try to

build such a model directly from the subsets of $c(P)$. In step 0 one constructs the set g_0 of those $Z \subseteq c(P)$ fulfilling the following conditions:

↓ page 12

$$\begin{aligned}
\neg R \in c(P) &\Rightarrow (\neg R \in Z \iff R \notin Z) \\
Q \wedge R \in c(P) &\Rightarrow (Q \wedge R \in Z \iff Q, R \in Z) \\
[a;b]R \in c(P) &\Rightarrow ([a;b]R \in Z \iff [a][b]R \in Z) \\
[a \cup b]R \in c(P) &\Rightarrow ([a \cup b]R \in Z \iff [a]R, [b]R \in Z) \\
[a^*]R \in c(P) &\Rightarrow ([a][a^*]R \in Z) \\
[Q?]R \in c(P) &\Rightarrow ([Q?]R \in Z \iff (Q \in Z \Rightarrow R \in Z))
\end{aligned}$$

For each atomic program A occurring in P we define $\pi_0(A)$ by

$$(Z, Z') \in \pi_0(A) \ :\iff \ ([A]R \in Z \Rightarrow R \in Z')$$

If after the i-th step there is a $Z \in g$ and a $[a]R \in c(P)$ such that $[a]R \notin Z$, but also

$$\text{for all } Z' \text{ such that } (Z, Z') \in \pi_i(a) \text{ we have } R \in Z',$$

then we define $g_{i+1} = g_i \setminus \{Z\}$ and $\pi_{i+1} := \pi_i \cap (g_{i+1} \times g_{i+1})$. After a maximum of $|g_0|$ steps we obtain a g_k for which there is no more such Z. If g_k is not empty, then we use $Z \in \nu_k(p) :\iff p \in Z$ to define the Kripke model $\mu = (g_k, \nu_k, \pi_k)$ and obtain

$$R \in c(P) \ \Rightarrow \ (Z \Vdash^{\mu} R \iff R \in Z).$$

Hence if a $Z \in g_k$ contains the formula P, then P is satisfiable. In particular one can show that P is satisfiable if and only if there exists a $Z \in g_k$ with $P \in Z$. Hence the construction of g_k above provides a decision procedure.

Other works ([1] and [13]) provide decision procedures for *deterministic propositional dynamic logic* DPDL. It interprets formulas and programs only over *deterministic* Kripke models; these are Kripke models (g, ν, π) in which each $\pi(A)$ is a partial function. For each $S \in g$ there is thus at most one $T \in g$ such that $S \xrightarrow{A} T$. Not every satisfiable PDL formula is also satisfiable in such a model: The formula $\neg[A]p \wedge \neg[A]\neg p$ for example is only satisfied in a state S for which there exist two states $S \xrightarrow{A} T, T'$ such that $T \Vdash p$ and $F' \Vdash \neg p$.

As explained in VARDI, WOLPER [13], every formula which is satisfiable in a deterministic Kripke model has a so-called tree-model.

↓ page 13

This is a Kripke model (g, ν, π) where the structure $(g, \cup_{A \in \mathcal{P}} \pi(A))$ is a tree (see 1.4). Moreover it is shown that for each formula P one can construct an automaton A_P of size $c^{l(P)^2}$ for certain c, which accepts exactly the tree-models of P. The formula P is thus satisfiable exactly if A_P accepts any tree. Procedures to decide this problem are already known, we refer the interested reader to the bibliography in [13].

↓ page 14

1.4 Tableaus

In this section we define tableaus and introduce the main ingredients of tableau-calculi.

Definition 6. *The tuple $(g, \lhd)$ of a set g and a relation $\lhd$ over g is called a* structure. *The elements $s, t, \ldots$ of g are called* nodes*; a set $s_0, \ldots, s_n$ of nodes such that $s_0 \lhd \ldots \lhd s_n$ is called* path *(from s_0 to s_n).*
If there is a path from s to t, then we write $s \leq t$; the relation $\leq$ is thus the reflexive transitive closure of the relation $\lhd$. For $s \leq t$ and $s \neq t$ we write $s < t$. If $s < t$ then t is called successor *of s and s is called* predecessor *of t. In case there is no $u \in g$ such that $s < u < t$ we also call them* direct successor *and* direct predecessor*, respectively. A node without successors is called an* end node*.*

Definition 7. *A* tree *is a structure $(T, \lhd)$ with the following properties:*

(i) There is exactly one node $t_0 \in T$ without predecessors. This node is called the root *of $(T, \lhd)$.*

(ii) For each node $s \in T$ there is exactly one path from t_0 to s.

We call a tree $(T', \lhd')$ a subtree *of $(T, \lhd)$ iff $T' \subseteq T$ and $\lhd' = \lhd |_{T'}$, and if for $s, t \in T'$ such that $s < t$ also every node u with $s < u < t$ is also in T'. The* length *of a tree $(T, \lhd)$ is the number of nodes in the longest path in $(T, \lhd)$.*

Definition 8. *A* Tableau $\mathcal{T}$ *(over a set of formulas $\mathcal{F}_\mathcal{T}$) is a structure of the form $(T, \lhd, x, \mathcal{F}_\mathcal{T})$. Here $(T, \lhd)$ is a finite tree and x is a function mapping each node t to a finite set of formulas $x(t) \subseteq \mathcal{F}_\mathcal{T}$ satisfying the following condition:*

> *If $x(s)$ is satisfiable and s is not an end node, then $x(t)$ is satisfiable for at least one direct successor t of s.*

↓ page 15

We will not always strictly distinguish between a structure $\mathcal{T}$ and the set of nodes T and we will say that we have nodes $s, t, \ldots \in \mathcal{T}$. Moreover we will identify nodes with their sets of formulas, unless this would cause confusion.

We thus will talk about a tableau "with root X" or "a tableau for X" when the root of the tableau is mapped to the set of formulas X.

A *Tableau-Calculus* consists of two components:

- a set of *rules* which define how a tableau for X can be constructed, starting from the root X and creating new nodes step by step.
- an easy to check property of tableaus which implies that the root X is not satisfiable. Tableaus with this property are called *closed*.

A tableau-calculus is called *complete* iff the rules allow us to construct a closed tableau for every non-satisfiable set of formulas X. A complete calculus is a useful decision procedure if for every non-satisfiable X there is a closed tableau of a size not above a computable limit which only depends on $l(X) := \sum_{P \in X} l(P)$.

Before we present the tableau-calculus for PDL, we will give a complete calculus for the decision problem whether a formula from the set $\mathcal{F}(A)$ is satisfiable. This is essentially the calculus for the modal logic K from RAUTENBERG [12].

Definition 9. *A set of formulas X is* closed *iff we have $0 \in X$ or it contains a formula and its negation. The set is called* simple *if all formulas $P \in X$ are (negated) propositional variables or of the form $[A]R$ or $\neg[A]R$. Moreover, let $X_A := \{R \mid [A]R \in X\}$.*

↓ page 16

In the above we say that a formula P is *of the form* $[A]R$ if there are $A \in \mathcal{P}_0$ and $R \in \mathcal{F}$ such that $P = [A]R$.

Lemma 1. *A simple set of formulas X is satisfiable if and only if it is not closed and for all $\neg[A]R \in X$ also $X_A; \neg R$ is satisfiable.*

Proof. Let X be satisfiable. Then we have $S \Vdash^{\mu} X$ in a Kripke model $\mu = (g, \nu, \pi)$ and for each $\neg[A]R \in X$ there is a $S_{A,R} \in g$ with $S \xrightarrow{A} S_{A,R}$ and $S \Vdash X_A; \neg R$.
For the other direction, suppose that X is not closed, and that for each $\neg[A]R \in X$ there is a Kripke model $\mu_{A,R}$ which has a state $S_{A,R}$ such that $S_{A,R} \Vdash X_A; \neg R$. One now combines all $\mu_{A,R}$ with a new state S_0 to a new Kripke model. We assume w.l.o.g. that all $g_{A,R}$ are disjoint and that S_0 is not in any $g_{A,R}$. Let $g := \{S_0\} \cup \bigcup\{g_{A,R} \mid \neg[A]R \in X\}$. Let $\nu'(p) := \{S_0\}$ if $p \in X$ and $\nu'(p) := \varnothing$ otherwise (here we use that X is not closed). Let $\nu(p) := \nu'(p) \cup \bigcup\{\nu_{A,R}(p)\}$. Moreover, let $\pi'(A) := \{(S_0, S_{A,R}) \mid \neg[A]R \in X\}$

and $\pi(A) := \pi'(A) \cup \{\pi_{A,R} \mid \neg[A]R \in X\}$. In $\mu := (g, \nu, \pi)$ we then have $S_0 \xrightarrow{A} S_{A,R}$ for all $\neg[A]R \in X$. One quickly checks that $S_0 \Vdash^\mu X$. □

The rules of the calculus are as follows:

$$(\neg)\ \frac{X; \neg\neg P}{X; P} \qquad (\wedge)\ \frac{X; P \wedge Q}{X; P; Q} \qquad (\neg\wedge)\ \frac{X; \neg(P \wedge Q)}{X; \neg P \ \mid \ X; \neg Q}$$

$$(K)\ \frac{X; \neg[A]P}{X_A; \neg P}$$

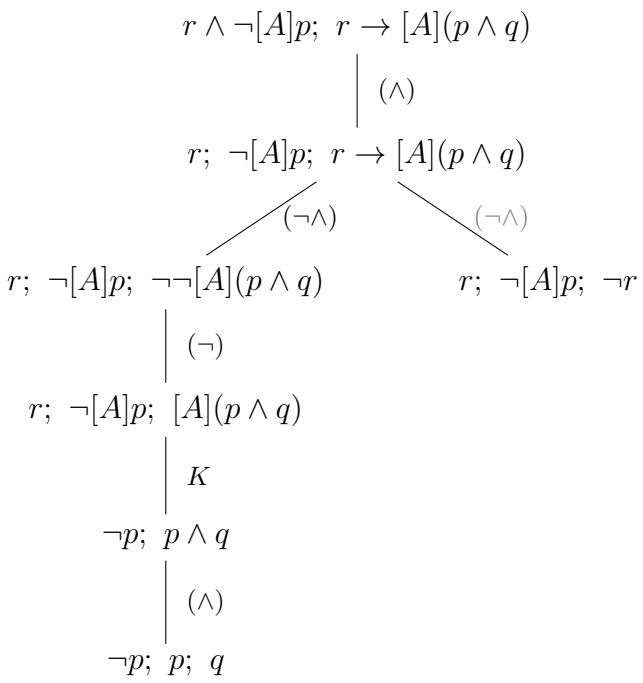
We use the same notation for rules as in [12]: Successors of a node Z can be generated using the rule $(\neg\wedge)$, if it is of the form $X; \neg(P \wedge Q)$, i.e. if there are $X \subseteq \mathcal{F}$ and $P, Q \in \mathcal{F}$ such that $Z = X; \neg(P \wedge Q)$. The node Z is then given two successors $X; \neg P$ and $X; \neg Q$. This procedure is called the *application* of the rule $(\neg\wedge)$ on Z (or also on $\neg(P \wedge Q)$ in Z). The application of other rules is analogously. To each node only one rule may be applied once. In the calculus we call a tableau $\mathcal{T}$ closed when all end nodes of $\mathcal{T}$ are closed.

↓ page 17

One constructs a closed tableau $\mathcal{T}$ for a non-satisfiable Z as follows: Let X be an already constructed, non-satisfiable node of the tableau which is not closed already. If X is not simple, then one of the rules $(\wedge)$, $(\neg\wedge)$ or $(\neg)$ can be applied to it. Then X is given one or two successors, and these nodes are also not satisfiable. If instead X is simple, then by Lemma 1 there is a $\neg[A]R \in X$ such that $X_A; \neg R$ is not satisfiable. One can then apply rule K to obtain exactly this node.

Each successor Y of a non-satisfiable node X constructed in this way is also non-satisfiable, and moreover we have $l(Y) < l(X)$. Finally one obtains a tableau which is not longer than $l(Z)$ and of which all satisfiable end nodes are closed.

Example 2. *We give a tableau for the set* $\{r \wedge \neg[A]p,\ r \to [A](p \wedge q)\}$. *After each node we indicate the rule which was used to construct its successors.*



↓ page 18

1.5 Local Tableaus

As we have seen in the example in the previous section, one can easily construct a tableau for any set of formulas $X \subseteq \mathcal{F}(A)$ such that all end nodes $X_1, \ldots, X_n$ are simple and

$$S \Vdash^\mu X \quad \Longleftrightarrow \quad S \Vdash^\mu X_i \text{ for some } i \leq n.$$

holds for any state S of any Kripke model μ. We now want to show how to construct a tableau with a similar property for any $X \subseteq \mathcal{F}$. To do so, we need to extend the set of "formulas" $\mathcal{F}_\mathcal{T}$, which are used in nodes of such tableaus, beyond $\mathcal{F}$.

Definition 10. *An n-formula is a string, which is obtained by replacing one or more program operators of the form $[a^*]$ by $[a^{(n)}]$. Let the function f map each n-formula to the formula $P \in \mathcal{F}$ from which it can be obtained in this way; and for all $P \in \mathcal{F}$, let $f(P) := P$. Let $\mathcal{F}^n$ be the set of all n-formulas. We also denote n-formulas with the letters $P, Q, \ldots$ and we also call them formulas. To stress that a formula is not an n-formula we also call it* normal. *We define for all $P \in \mathcal{F}^n$ $S \Vdash P :\Longleftrightarrow S \Vdash f(P)$. This implicitly defines when a set $X \subseteq \mathcal{F} \cup \mathcal{F}^n$ is satisfiable. From now on by "set of formulas" we always mean a (finite) set $X \subseteq \mathcal{F} \cup \mathcal{F}^n$. We call a node* normal *if it only contains normal formulas, otherwise we also call it an n-node. A node which contains a formula of the form $\neg[a^{(n)}]P$ we call a $\neg[a^{(n)}]$-node.*

Definition 11. *A* local tableau *is a tableau which at its root has a normal set of formulas, and which is constructed using the classical rules*

$$(\neg)\ \frac{X;\neg\neg P}{X;P} \qquad (\wedge)\ \frac{X;P\wedge Q}{X;P;Q} \qquad (\neg\wedge)\ \frac{X;\neg(P\wedge Q)}{X;\neg P \ \mid\ X;\neg Q}$$

↓ page 19

and the following rules:

$$(\neg\cup)\ \frac{X;\neg[a\cup b]P}{X;\neg[a]P \ \mid\ X;\neg[b]P} \qquad (\neg?)\ \frac{X;\neg[Q?]P}{X;Q;\neg P} \qquad (\neg;)\ \frac{X;\neg[a;b]P}{X;\neg[a][b]P}$$

$$(\cup)\ \frac{X;[a\cup b]P}{X;[a]P;[b]P} \qquad (?)\ \frac{X;[Q?]P}{x;\neg Q \ \mid\ X;P} \qquad (;)\ \frac{X;[a;b]P}{X;[a][b]P}$$

$$(\neg n)\ \frac{X;\neg[a^*]P}{X;\neg P \ \mid\ X;\neg[a][a^{(n)}]P} \qquad (n)\ \frac{X;[a^*]P}{X;P;[a][a^{(n)}]P}$$

In addition, the following extra conditions must be obeyed when constructing local tableaus:

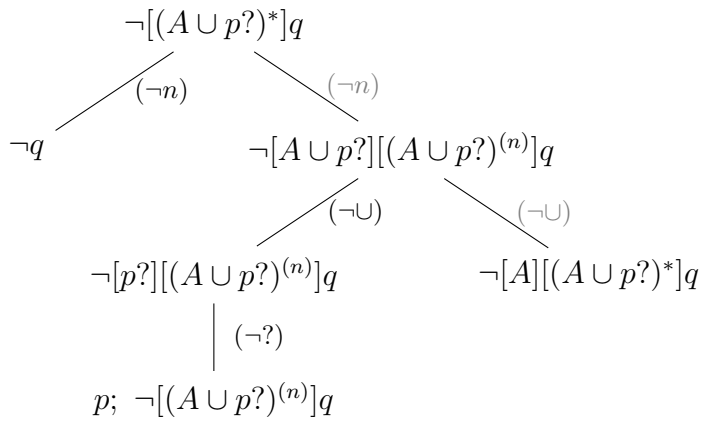
1. *Instead of a node $X;\neg[A]P$ or $X;[A]P$ with an n-formula P we always obtain the node $X;\neg[A]f(P)$ or $X;[A]f(P)$, respectively.*
   That is, if an atom A is reached, switch back from (n) to $*$ in P.
2. *Instead of a node $X;[a^{(n)}]P$ we always obtain the node X.*
3. *If it is possible, then a rule must be applied to an n-formula.*
4. *No rule may be applied to a $\neg[a^{(n)}]$-node.*

If we start with a set X, and application of one of the rules yields the sets $X_1,\ldots,X_n$ $(n\leq 2)$, then also here we always have $S \Vdash^\mu X \iff S \Vdash^\mu X_i$ for some $i\leq n$. Moreover, when a rule is applied to any formula then it is replaced by a formula which is simpler in a certain sense; for example, formulas are shorter or start with a simpler program operator. A normal set, to which no more rules can be applied, is simple.

Definition 12. *A local tableau is called* maximal*, iff no more rules can be applied to any of its end nodes.*

Wherever an n-node occurs in a tableau, then it is located in a sub-tableau which consists only of n-nodes, and which has end nodes that are $\neg[a^{(n)}]$-nodes or normal successors due to extra conditions 1 or 2 in Definition 11. We give an example.

↓ page 20



Lemma 2. *For every X there is a maximal local tableau of length not greater than $l(X)$.*

Proof. We first define in parallel for each program a and each formula P its *measure* m:

$$\begin{aligned}
m(A) &:= 0\\
m(a^{(n)}) &:= 0\\
m(Q?) &:= \max(m(Q), m(\neg Q)) + 1\\
m(a;b) &:= m(a) + m(b) + 1\\
m(a \cup b) &:= m(a) + m(b) + 1\\
m(a^*) &:= 1 + m(a)
\end{aligned}$$

$$\begin{aligned}
m(0) &:= 0\\
m(p) &:= 0\\
m(\neg p) &:= 0\\
m(\neg\neg P) &:= 1 + m(P)\\
m(P \wedge Q) &:= 1 + m(P) + m(Q)\\
m(\neg(P \wedge Q)) &:= 1 + m(\neg P) + m(\neg Q)\\
m([a]P) &:= m(a) + m(P)\\
m(\neg[a]P) &:= m(a) + m(\neg P)
\end{aligned}$$

By induction one shows that the numbers $m(a)$ (and $m(P)$) are upper bounds for the number of different formulas which can appear in a local tableau for $[a]0$ or $\neg[a]0$ (and for P, respectively) and to which a rule can be applied. We note that in each application of a rule a formula with a lower measure is generated. One further shows by induction that $m(a) < l([a])$ for all programs a and $m(P) < l(P)$ for all formulas P. It immediately follows that a local tableau for X contains less than $l(X)$ different formulas to which a rule can be applied.

↓ page 21

Let now $X_1 \lhd \ldots \lhd X_n$ be a path of a local tableau $\mathcal{T}$, which is constructed in such a way that (adhering to extra condition 3) in each node X_i $(i < n)$ a formula P_i with maximal $m(P_i)$ is getting replaced. We now show:

$$\text{For } j > i \text{ we have } P_i \notin X_j.$$

Suppose P_i is normal. We have $m(P_j) \leq m(P_i)$ for all $j \geq i$. No P_j can yield a P_i, hence $P_i \notin X_j$ for $j > i$.

Suppose P_i is an n-formula and let X_l be the first normal node after X_i. For $i \leq j < l$ we also have $m(P_j) \leq m(P_i)$, hence $P_i \notin X_j$ for $i < j \leq l$. If P_i is a non-negated n-formula, then P_i is of the form $[a_1] \ldots [a_n][a^{(n)}]Q$ with a normal Q, and we have $P_k = [a^*]Q$ for some $k < i$. Because $[a^*]Q \notin X_j$ holds for $j > k$, and P_i can only be generated from $[a^*]Q$, we also have $P_i \notin X_j$ for all $j > l$. The analogue claims hold for negated n-formulas.

Along the path no formula is replaced more than once, and no more than $m(X_1)$ many formulas occur, hence it cannot be longer than $l(X_1)$. □

Recall from Definition 8 that $x(t)$ denotes the set of formulas at node t.

Lemma 3. *If s and t are nodes of a local tableau with $s < t$, then $x(s) \neq x(t)$.*

Proof. When a rule is applied any formula P is replaced by (maximally) two formulas P_1, P_2. Because $m(P) > m(P_i)$ $(i = 1, 2)$ using the measure $m'(X) := \sum_{P \in X} 3^{m(P)}$ we always have $m'(x(s)) > m'(x(t))$. □

Lemma 4. *Let μ be a Kripke model over g, and let $S \in g$. Let $X = X'; \neg[a]P$ be a node of a local tableau with $S \Vdash X$. Let X' be normal and let P be normal if a is atomic; otherwise let P be n-formula. Moreover let $T \in g$ with $S \xrightarrow{a} T$ and $T \Vdash \neg P$.*

Then there is a normal node $Y \geq X$ with $S \Vdash Y$, containing the formula $\neg[a_1] \ldots [a_n] f(P)$ for some $n \in \omega$.

If $n > 0$, then a_1 is an atomic program and we have $S \xrightarrow{a_1;\ldots;a_n} T$. If $n = 0$ then $S = T$.

Proof. By induction over a.

$a = A$: The claim holds with $Y = X$.

$a = Q?$: There is only one state T with $S \xrightarrow{Q?} T$, namely $T = S$. The immediate successor of X is $X'; Q; \neg f(P)$, it is normal and holds in S.

↓ page 22

$a = b \cup c$**:** We have $S \xrightarrow{b} T$ or $S \xrightarrow{c} T$. Suppose w.l.o.g. that $S \xrightarrow{b} T$. Then the successor $X; \neg[b]P$ of X holds in S. By induction hypothesis we get the claim.

$a = b; c$**:** There is a $U \in g$ such that $S \xrightarrow{b} U \xrightarrow{c} T$. In U we have $\neg[c]P$ and thus we can apply the induction hypothesis to the successor $X'; \neg[b][c]P$. Hence there is a $Z \geq X$ with $S \Vdash Z$ and Z contains the n-formula $\neg[a_1] \dots [a_n][c]P$. If $n > 0$ then we are done, otherwise we again apply the induction hypothesis to Z.

$a = b^*$**:** If $S = T$, then the successor $X'; \neg P$ holds in S. If $S \neq T$, then there exists a $U \in g$ such that $U \neq S$ and $S \xrightarrow{b} U \xrightarrow{b^*} T$ and $U \Vdash \neg[b^*]P$. Hence we can apply the induction hypothesis to the successor $X; \neg[b][b^{(n)}]P$. Because $S \neq U$ there exists a successor of this node which holds in S and which contains an n-formula of the form $\neg[a_1] \dots [a_n][b^{(n)}]P$ where a_1 is atomic.

□

Lemma 5. *Let $\mathcal{T}$ be a local tableau in which each end node is normal or a $\neg[a^{(n)}]$-node. Let $X_1, \dots, X_n$ be the normal end nodes. Let S be a state of a Kripke model μ over g.*

We have

$$S \Vdash^\mu X \quad \Longleftrightarrow \quad S \Vdash^\mu X_i \text{ for some } i \leq n$$

Proof. Right to left: From $Y \lhd Z$ and $S \Vdash^\mu Z$ we get $S \Vdash^\mu Y$, and thus from $S \Vdash^\mu X_i$ we immediately get $S \Vdash^\mu X$.

For the other direction (left to right) we show that for each normal node $Y \in \mathcal{T}$ with $S \Vdash^\mu Y$ there exists a normal node $Z > Y$ with $S \Vdash^\mu Z$. We only need to consider the two cases in which not all immediate successors of Y are normal:

- To each node which contains a non-negated n-formula a rule can be applied. If the successor of Y is obtained using (n), one can thus find a path of nodes that are satisfied in S, leading from Y to a normal node Z.
- Suppose $Y = Y'; \neg[a^*]P$, and the successors of Y are obtained by replacing $\neg[a^*]P$ according to (n). If a is atomic, then already both successors of Y are normal. Hence suppose a is not atomic. Then there is a state $U \in g$ such that $S \xrightarrow{a^*} U$ and $U \Vdash^\mu \neg P$: If $S = U$, then already the successor $Y'; \neg P$ of Y is satisfied in S and normal. If $S \neq U$,

↓ page 23

then there exists a T with $T \neq S$ and $S \xrightarrow{a} T \xrightarrow{a^*} U$ and $T \Vdash \neg[a^*]P$. Applying Lemma 4 to $Y; \neg[a][a^{(n)}]P$ and T gives us a node $Z > Y$ which is satisfied in S.

□

↓ page 24

1.6 PDL-Tableaus

Suppose the set of formulas X contains a formula of the form $\neg[a_1]\dots[a_n]P$. A necessary condition for $S \Vdash X$ to hold in a Kripke over g, is that there exist states $T_1, \dots, T_n \in g$ such that $S \xrightarrow{a_1} T_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} T_n$ and $T_n \Vdash \neg P$.

In this section we provide rules for *PDL*-tableaus with which this condition about X can be checked. This includes a rule which *marks* a formula of the form $\neg[a_1]\dots[a_n]P$ with the formula P. We then write $\neg[a_1]\dots[a_n]P^P$. If a set of formulas contains a marked formula we call it *loaded*, otherwise *free*. Analogously we will talk about loaded and free nodes.

The rules of PDL-tableaus are those of local tableaus together with the following:

$$(M+)\ \frac{X; \neg[a_0]\dots[a_n]P}{X; \neg[a_0]\dots[a_n]P^P}\quad X \text{ free} \qquad \text{the } \textit{loading rule},$$

$$(M-)\ \frac{X; \neg[a]P^R}{X; \neg[a]P} \qquad \text{the } \textit{liberation rule},$$

$$(At)\ \frac{X; \neg[A]P^R}{X_A; \neg P^{R\backslash P}} \qquad \text{the } \textit{critical} \text{ rule}.$$

In the above we use $\neg P^{R\backslash P} := \begin{cases} \neg P & \text{if } R = P \\ \neg P^R & \text{otherwise} \end{cases}$

The rules $(\neg\cup)$, $(\neg n)$, $(\neg;)$ and $(\neg?)$ for local tableaus should also be applicable to marked formulas. The generated formulas then have to be marked in a specific way which is yet to be determined. We thus state these rules again for applications to marked formulas.

$$(\neg\cup)\ \frac{X; \neg[a \cup b]P^R}{X; \neg[a]P^R \quad | \quad X; \neg[b]P^R} \qquad (\neg;)\ \frac{X; \neg[a;b]P^R}{X; \neg[a][b]P^R}$$

$$(\neg n)\ \frac{X; \neg[a^*]P^R}{X; \neg P^{R\backslash P} \quad | \quad X; \neg[a][a^{(n)}]P^R} \qquad (\neg?)\ \frac{X; \neg[Q?]P^R}{X; Q; \neg P^{R\backslash P}}$$

↓ page 25

Note that when applying the rules (At), $(\neg n)$ and $(\neg ?)$ to a marked formula, the marking disappears if and only if the resulting formula is the negation of the mark.

Definition 13. *Let $X_1 \lhd \ldots \lhd X_n$ be a path of a tableau constructed using the rules defined so far. If no X_i $(i > 1)$ is obtained using (At), then we call the path* uncritical*, otherwise* critical*. If all X_i are loaded, then we also call the path loaded.*

Definition 14. *A* PDL-tableau *is a tableau constructed using all the rules defined so far, which in addition to the extra conditions 1 to 4 also adheres to the following extra conditions and restrictions:*

5. *To a node obtained using $(M+)$ we may not apply $(M-)$.*
6. *A normal node $X = x(t)$ for which there is an $s < t$ with $x(s) = x(t)$, is an end node, if the path from s to t is critical, and when the path from s to t is loaded, if X is loaded.*
7. *Every loaded node that is not an end node by condition 6, has a successor.*

A local *PDL-tableau is a PDL-tableau in which no node is obtained using (At).*

Each PDL-tableau $\mathcal{T}$ can in a natural way be partitioned into local PDL-tableaus $\mathcal{T}_1, \ldots, \mathcal{T}_n$, namely into the maximal local sub-tableaus. From each normal end node of a $\mathcal{T}_i$ which is not already an end node of $\mathcal{T}$ we obtain the root of a $\mathcal{T}_j \neq \mathcal{T}_i$ using (At). A local PDL-tableau only differs from a local tableau in possible applications of $(M+)$ and $(M-)$, so they have mostly the same properties. We therefore leave out the addition PDL where it does not lead to misunderstandings. We also just talk about *tableaus* when we mean PDL-tableaus.

↓ page 26

Lemma 6. *There is a maximum length for tableaus for X.*

Proof. Similar to Lemma 2 one first shows that a tableau $\mathcal{T}$ for X cannot contain more than $n := l(x)$ different non-marked formulas. Moreover, it follows from Lemma 3 for all nodes $s < t \in \mathcal{T}$ where $x(s) = x(t)$ that the path from s to t is critical. Hence no path contains more than 2^n free normal nodes.

Furthermore in $\mathcal{T}$ there cannot be more than $n \cdot 2^n$ different loaded sets for each possible marking. A loaded path thus contains at most $n \cdot 2^n$ normal nodes. If there are only n-nodes between two such nodes, then there are less than 2^n nodes, because they are then on the same uncritical path. A loaded

path is thus not longer than $n \cdot 2^{2n}$. Between two loaded paths there is at least one free node, hence only less than 2^n loaded paths can follow after each other. The whole tableau $\mathcal{T}$ is thus no longer than $n \cdot 2^{4n}$. □

Definition 15. *If $\mathcal{T}$ is a tableau, then we define for $s, t \in \mathcal{T}$:*

$$s \lhd' t \quad :\Longleftrightarrow \quad s \lhd t, \text{ or } s \text{ is an end node, and there exists a } u < s \text{ with } u \lhd t \text{ and } x(u) = x(s), \text{ from which a loaded and critical path leads to } s.$$

Moreover, let $\leq'$ denote the reflexive transitive closure of $\lhd'$. The relation is in general not a partial order (i.e. it might not be anti-symmetric). However, for free nodes X, Y we have $X < Y \iff X <' Y$. A sequence $t_1 \lhd' \ldots \lhd' t_n$ is called a $\lhd'$-path.

Lemma 7. *Let S be a state of a Kripke model over g, and let $X = X'; \neg[a]P^R$ be a normal node true in S which is part of tableau $\mathcal{T}$ in which no node is obtained using $(M-)$. Moreover, let $T \in g$ with $S \xrightarrow{a} T$ and $T \Vdash \neg P$. Then there is a $\lhd'$-path of satisfiable nodes from X to a normal node $Y = Y'; \neg P^{R \setminus P}$ which holds in T.*

Proof. We first note that in $\mathcal{T}$ every loaded node s has successors with respect to $\lhd'$, obtained by applying a rule to $x(s)$. Then there exists a $\lhd'$-path of satisfiable nodes from X to a node $Z = Z'; \neg[a]P^R$ which is true in S, and whose successors are obtained by replacing the marked formula. The remainder of the proof is by induction on a.

↓ page 27

Base case:

$a = A$**:** Let $T \in g$ with $S \xrightarrow{A} T$ and $\Vdash \neg P$. The $\lhd'$-successor $Z'_A; \neg P^{R \setminus P}$ of Z is true at this T and is of the required form.

$a = Q?$**:** We have $T = S$. The $\lhd'$-successor $Z'; Q; \neg P^{R \setminus P}$ of Z is thus true in T and is of the required form.

Induction hypothesis: Assume now that the claim holds for programs b and c.

Induction step:

$a = b \cup c$ Let $S \xrightarrow{b \cup c} T$ and $T \Vdash \neg P$. Then $S \xrightarrow{b} T$ or $S \xrightarrow{c} T$ and one of the $\lhd'$-successors $Z'; \neg[b]P^R$ and $Z'; \neg[c]P^R$ of Z holds at S. Applying the induction hypothesis to this node gives the claim.

$a = b; c$ For $S \xrightarrow{b;c} T$ there exists a U with $S \xrightarrow{b} U \xrightarrow{c} T$ and $U \Vdash \neg[c]P$. Applying the induction hypothesis to the $\lhd'$-successor $Z'; \neg[b][c]P^R$ of

Z and U yields a node $W = W'; \neg[c]P^R$ with $Z <' W$, which holds in U. Applying the induction hypothesis to W and T gives the claim.

$a = b^*$ If $S \xrightarrow{a^*} T$, then there is a minimal n such that $S \xrightarrow{a^n} T$. We prove this case by induction over n.

Base case $n = 0$: If $n = 0$, then $S = T$, the $\lhd'$-successor $Z'; \neg P^{R \setminus P}$ holds in S, and it is of the required form.

Induction hypothesis: Suppose the claim is proven for $m = n - 1$.

Induction step: If $n > 0$, then there exists a $U \in g$ with $U \neq S$ and $S \xrightarrow{b} U \xrightarrow{b^m} T$. The $\lhd'$-successor $Z'; \neg[b][b^{(n)}]P$ holds in S. By Lemma 4 there is a normal node $W \geq' Z$ with $W = W'; \neg[a_1] \ldots [a_k][b^*]P^R$ such that $S \xrightarrow{a_1;\ldots;a_k} U$ and which holds in S. All a_i have lower complexity than b^*, applying k-many times the hypothesis of the induction over the program structure yields the existence of a node $V >' W$ with $V = V'; \neg[b^*]P^R$ which holds in U. Now $U \xrightarrow{b^m} T$, applying the hypothesis of the induction over n gives the claim. □

↓ page 28

Lemma 8. *Let $\mathcal{T}$ be a tableau and let $Y \in \mathcal{T}$ be a free satisfiable normal node. If Y is not an end node, then there is a free satisfiable normal node $Z > Y$.*

Proof. If $(M+)$ is not applied to Y, then this is the statement of Lemma 4. If $(M+)$ is applied to Y, then the successor of Y is a normal node of the form $Y'; \neg[a_1] \ldots [a_n]P^P$. With Lemma 7 one finds a $\lhd'$-path of satisfiable nodes to a normal free node Z which is of the form $Z'; \neg P^{P \setminus P}$, or which is obtained by applying $(M-)$. In both cases is Z a satisfiable normal free node with $Z > Y$. □

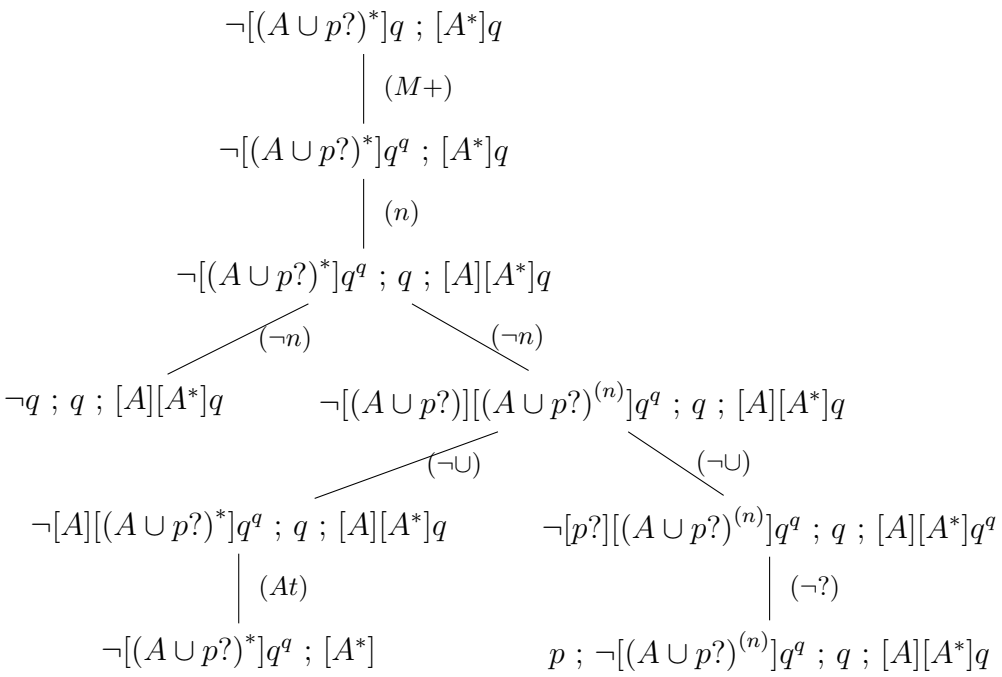
↓ page 29

1.7 The PDL-Calculus

Definition 16. *A tableau $\mathcal{T}$ is called* closed *when all normal free end nodes of $\mathcal{T}$ are closed.*

In the remainder of Part 1 we show that the PDL-Calculus we defined in the previous sections is complete; that is, any set of formulas $X \subset \mathcal{F}$ is satisfiable if and only if there is no closed tableau for X.

However, before that we give an example of a closed tableau for the set $X = \{\neg[(A \cup p?)^*]q, [A^*]q\}$.



This tableau has three end nodes, whose properties we shall discuss briefly. Recall that 'normal' means 'no (n)' and 'free' means 'no markings'.

- The end node $\{\neg q\ ;\ q\ ;\ [A][A^*]q\}$ is closed. It is normal and free, and all such end nodes of a closed tableau are closed.

↓ page 30

- The end node $\{\neg[(A \cup p?)^*]q^q\ ;\ [A^*]\}$ is not closed. But it is loaded, and it has a predecessor which has the same set (of formulas), and moreover all nodes between them are loaded. Hence by condition 6 it is an end node.
- The end node $\{p\ ;\ \neg[(A \cup p?)^{(n)}]q\ ;\ q\ ;\ [A][A^*]q^q\}$ is a $\neg[a^{(n)}]$-node. No rule may be applied to it.

We note that each end node of a closed tableau has at least one of these properties.

Definition 17. *A normal set of formulas $X \subseteq \mathcal{F}$ is called* inconsistent *if there exists a closed tableau for X. Otherwise X is called* consistent*.*

Theorem 2 (Correctness Theorem)**.** *Any satisfiable set of formulas $X \subseteq \mathcal{F}$ is consistent.*

Proof. It follows immediately from Lemma 8 that any tableau of a satisfiable set of formulas X has a free satisfiable normal end node. In particular this node is open. Hence X is consistent. □

To show the other direction, i.e. that any consistent set of formulas is also satisfiable, we will show how to use the open tableaus for X to construct a model for X. For this we define a certain class of Kripke models, the class of *model graphs*.

↓ page 31

1.8 Model Graphs

Definition 18. *A set of formulas X is called* saturated *if it fulfils the following conditions:*

$$
\begin{array}{lcl}
\neg\neg P \in X & \Rightarrow & P \in X \\
P \wedge Q \in X & \Rightarrow & P \in X \text{ and } Q \in X \\
\neg(P \wedge Q) \in X & \Rightarrow & \neg P \in X \text{ or } \neg Q \in X \\
[a;b]P \in X & \Rightarrow & [a][b]P \in X \\
[a \cup b]P \in X & \Rightarrow & [a]P \in X \text{ and } [b]P \in X \\
[Q?]P \in X & \Rightarrow & \neg Q \in X \text{ or } P \in X \\
[a^*]P \in X & \Rightarrow & P \in X \text{ and } [a][a^*]P \in X \\
\neg[a;b]P \in X & \Rightarrow & \neg[a][b]P \in X \\
\neg[a \cup b]P \in X & \Rightarrow & \neg[a]P \in X \text{ or } \neg[b]P \in X \\
\neg[Q?]P \in X & \Rightarrow & Q \in X \text{ and } \neg P \in X \\
\neg[a^*]P \in X & \Rightarrow & \neg P \in X \text{ or } \neg[a][a^*]P \in X
\end{array}
$$

Definition 19. *A* model graph *is a Kripke model $\mu = (g, \pi, \nu)$ fulfilling the conditions (i) to (iv).*

(i) Each element of g is a saturated set of formulas X with $0 \notin X$ and $p \in X \Rightarrow \neg p \notin X$.

(ii) *If $X \in g$ and $p \in X$, then $X \in \nu(p)$.*
This should be: "$p \in X$ iff $X \in \nu(p)$.", see Definition 4.18 in [14].

(iii) *If $(X, Y) \in \pi(A)$ and $[A]P \in X$, then $P \in Y$.*

(iv) *If $X \in g$ and $\neg[a]P \in X$, then there exists a $Y \in g$ such that $(X, Y) \in \pi'(a)$ and $\neg P \in Y$.*

Here π' is a function on $\mathcal{P}$, which for $\mathcal{P}_0$ agrees with π, for tests is defined by

$$\pi'(P?) := \{(X, X) \mid X \in g \text{ and } P \in X\}$$

and for programs of the form $a;b$, $a \cup b$ and a^* is defined inductively like π.

For each Kripke model one can easily construct a model graph:

> For $S \in g$ let $X_S := \{P \in \mathcal{F} \mid S \models^\mu P\}$, let $g' := \{X_s \mid S \in g\}$.
> Moreover, let $\pi'(A) := \{(X_S, X_T) \mid X_S \ {}_A\!\subseteq X_T\}$ for all $A \in \mathcal{P}_0$
> and $\nu'(p) := \{X_S \mid p \in X_S\}$ for all $p \in \mathcal{F}_0$.

It is easy to show that $\mu' := (g', \pi', \nu')$ fulfils conditions (i) to (iv).

↓ page 32

Lemma 9. *If $\mu = (g, \pi, \nu)$ is a model graph and $X \in g$ and $P \in X$, then we have $X \Vdash^\mu P$.*

Proof. We show by simultaneous induction over the structure of formulas and programs, that

$(+)$ $\quad P \in X \ \Rightarrow\ X \Vdash P$ (with $\Vdash = \Vdash^\mu$)

$(-)$ $\neg P \in X \ \Rightarrow\ X \nVdash P$

for all formulas P and

(0) $[a]P \in X \ \Rightarrow$ if $(X, Y) \in \pi(a)$, then $P \in Y$

holds for all programs a (and all formulas P).

If $P = 0$ or P is a propositional variable, then $(+)$ and $(-)$ hold because of the conditions (i) and (ii).

Let thus P be a formula such that $(+)$ and $(-)$ are already proven for its subformulas. Moreover, suppose (0) is already proven for all programs occurring in P.

Suppose $P = \neg Q$. If $P \in X$, then by $(-)$ and the induction hypothesis we have $X \nVdash Q$, hence $X \Vdash \neg Q = P$, and thus $(+)$. If $\neg P = \neg\neg Q \in X$, then because X is saturated we have $Q \in X$ and $X \Vdash Q$ by induction hypothesis. But then we have $X \nVdash \neg Q = P$, and this is $(-)$.

Suppose $P = Q \wedge R$. If $P \in X$, then $Q \in X$ and $R \in X$, which by induction hypothesis implies $X \Vdash Q$ and $X \Vdash R$, thus $X \Vdash Q \wedge R = P$, hence $(+)$ holds. If $\neg P \in X$, then $\neg Q \in X$ or $\neg R \in X$, which implies $X \nVdash Q \wedge R = P$, which is $(-)$.

Suppose $P = [a]Q$. If $P \in X$, then by induction hypothesis for a we have that for all Y such that $(X, Y) \in \pi(a)$ we already have $Q \in Y$. By induction hypothesis for P we also have that $Q \in Y$ implies $Y \Vdash Q$, hence we have $X \Vdash [a]Q$. If $\neg P \in X$, then for every test which appears in a we already have $(+)$. From $(X, Y) \in \pi'(R?)$ we thus get $(X, Y) \in \pi(R?)$, and thus $(X, Y) \in \pi'(a)$ implies $(X, Y) \in \pi(a)$. Hence by condition (iv) there exists a Y such that $(X, Y) \in \pi(a)$ and $\neg Q \in Y$. By induction hypothesis we have $Y \Vdash \neg Q$, and thus $X \nVdash [a]Q$.

For atomic programs (0) always holds by condition (iii). Let thus $[a]P \in X$ and a a non-atomic program such that (0) is already proven for all its subprograms. Moreover, suppose (0) is already proven for all tests which occur in a.

↓ page 33

Suppose $a = R?$. Because X is saturated we have $\neg R \in X$ or $P \in X$. If $(X, Y) \in \pi(R?)$, then $Y = X$ and $X \Vdash R$. By $(-)$ we then have $\neg R \in X$ and thus $P \in X = Y$.

Suppose $a = b \cup c$. Because X is saturated we have $[b]P \in X$ and $[c]P \in X$. If now $(X, Y) \in \pi(a)$, then $(X, Y) \in \pi(b)$ or $(X, Y) \in \pi(c)$. In both cases by induction hypothesis we get $P \in Y$.

Suppose $a = b; c$. We have $[b][c]P \in X$. If $(X, Y) \in \pi(a)$, then there exists a $Z \in g$ such that $(X, Z) \in \pi(b)$ and $(Z, Y) \in \pi(c)$. By induction hypothesis we have $[c]P \in Z$ and $P \in Y$.

Suppose $a = b^*$. If $(X, Y) \in \pi(b^*)$, then there is a minimal $n \in \omega$ such that $(X, Y) \in \pi(b^n)$. We now prove (0) by induction over n. The case $n = 0$ is easy, because then $X = Y$ and because X is saturated we already have $P \in X$. Thus, suppose that that claim is proven for $m = n - 1$. We have $[b][b^*]P \in X$ and there is a $Z \in g$ such that $(X, Z) \in \pi(b)$ and $(Z, Y) \in \pi(b^{n-1})$. By induction hypothesis for a it follows that $[b^*]P \in Z$, and by induction hypothesis for n we get $P \in Y$. □

↓ page 34

1.9 Consistent Nodes

To construct model graphs from tableaus, we define the following relations for each tableau.

Definition 20. *Let $X < Y$ be nodes of a tableau. We write*

- $X \xrightarrow{A} Y$, *if the critical rule is applied to exactly one node Z with $X \leq Z < Y$ and to a formula of the form $\neg[A]P^R$.*
- $X \xrightarrow{P?} Y$, *if the path from X to Y is not critical and there exists a Z with $X \leq Z \leq Y$ and $P \in Z$.*
- $X \xrightarrow{a \cup b} Y$, *if $X \xrightarrow{a} Y$ or $X \xrightarrow{b} Y$.*
- $X \xrightarrow{a;b} Y$, *if there exists a node Z with $X < Z < Y$, such that $X \xrightarrow{a} Z$ and $Y \xrightarrow{b} Y$.*
- $X \xrightarrow{a^*} Y$, *if the path from X to Y is not critical, or if $X \xrightarrow{a^i} Y$ for some $i \geq 1$.*

Moreover, if $X < Y$ are such that Y is free and there is no free node Z such that $X < Z < Y$, then we call Y a first free successor *of X.*

We also need a few results about occurrences of consistent nodes in tableaus.

Lemma 10. *Every node $X = X'; \neg[a_1] \dots [a_n]P^P$ obtained using $(M+)$, has a free normal successor Y. If Y is a first free normal successor of a node $X = X'; \neg[a_1] \dots [a_n]P^P$, and is Y not obtained using $(M-)$, then Y is of the form $Y = Y'; \neg P$, and we have $X \xrightarrow{a_1} \dots \xrightarrow{a_n} Y$.*

Proof. By induction over n and the construction of the a_i. □

It follows that in particular every free normal node of a tableau which is not already an end node, has at least one free normal successor.

↓ page 35

Lemma 11. *Let $X \in \mathcal{T}$ be a free normal node with the first free normal successors $Y_1, \dots, Y_n$ ($n \geq 1$). If all Y_i are inconsistent, then also X is inconsistent.*

Proof. For each i let $\mathcal{T}_i$ be a closed tableau for Y_i.

If X is a node of a tableau $\mathcal{T}_i$, then the sub-tableau of $\mathcal{T}_i$ consisting of all nodes $Y \in \mathcal{T}_i$ such that $X \leq Y$, is a closed tableau for X.

If X is not in any $\mathcal{T}_i$ node, then we build a new tableau for X by "cutting" $\mathcal{T}$ at the nodes Y_i and "stapling" onto them the corresponding $\mathcal{T}_i$. The new tableau is a closed tableau for X.

In both cases it follows that X is inconsistent. □

Definition 21. *A loaded normal set of formulas X is called* inconsistent *if $X^- := \{P \mid P \in X \text{ or } P^R \in X \text{ for a certain } R\}$ is inconsistent, otherwise* consistent.

Lemma 12. *If X is a free normal consistent node in a tableau $\mathcal{T}$, then there is a first free normal consistent successor $Y > X$ such that every normal node on the path from X to Y is consistent.*

Proof. Let $Y_1, \ldots, Y_n$ be the first free normal consistent successors of X (Lemma 11 guarantees that they exist). Towards a contradiction, assume that for each i there is a normal inconsistent node Z_i with $X < Z_i < Y_i$. These nodes are all loaded. By application of $(M-)$ to the Z_i one obtains a tableau in which all first free normal successors of X are inconsistent. By Lemma 11 then X must be inconsistent; hence our assumption contradicts the premise. □

↓ page 36

1.10 The Completeness Theorem

Definition 22. *We call a tableau $\mathcal{T}$ saturating, if the critical rule is only applied to simple nodes.*

Theorem 3 (Model Existence)**.** *If Z_0 is a normal consistent set of formulas, then there exists a model graph $\mu = (g, \nu, \pi)$ and an $S \in g$ with $Z_0 \subseteq S$.*

Proof. Let $\mathcal{T}_0$ be a maximal local tableau for Z_0, in which $(M+)$ is not used. Moreover, let M_0 be the smallest set of tableaus which

(a) contains $\mathcal{T}_0$ and

(b) for any simple consistent node Y of any $\mathcal{T} \in M_0$ also contains every maximal saturating tableau for Y^-

For nodes $X, Y \in \mathcal{T} \in M_0$, let

$$\begin{aligned} S_{X,Y} &:= \{f(P) \mid \text{there exists a } Z \text{ such that } X \leq Z \leq Y \text{ and } P \in Z^-\}, \\ g &:= \{S_{X,Y} \mid X \text{ and } Y \text{ are consistent roots and end nodes of a} \\ &\qquad \text{local sub-tableau of some } \mathcal{T} \in M_0 \text{ and } Y \text{ is simple.}\} \end{aligned}$$

Each $S_{X,Y}$ is a saturated set of formulas and fulfils condition (i) for model graphs. We note that $|g| \leq 2^{l(X)}$. Moreover, for each $S, T \in g$ let

$$\begin{aligned} S \in \nu(p) &:\Longleftrightarrow p \in S \quad \text{for all } p \in \mathcal{F}_0 \\ (S,T) \in \pi(A) &:\Longleftrightarrow S_A \subseteq T \quad \text{for all } A \in \mathcal{P}_0 \end{aligned}$$

With these definitions $\mu = (g, \nu, \pi)$ fulfils the conditions (ii) and (iii) for model graphs. By induction over a one easily shows that:

If $X, Y, X', Y' \in \mathcal{T} \in M_0$ with $S_{X,Y} \in g$ and $S_{X',Y'} \in g$, and Z, Z' are nodes with $X \leq Z \leq Y$, $X' \leq Z' \leq Y'$ and $Z \xrightarrow{a} Z'$, then we have $(S_{X,Y}, S_{X',Y'}) \in \pi'(a)$.

We now show that μ also fulfils condition (iv). For this, let $S \in g$ and $\neg[a]P \in S$. We have $S = S_{X,Y}$ for some nodes X, Y of some $\mathcal{T} \in M_0$. Let $X = X_1 \lhd \ldots \lhd X_n = Y$ be the path from X to Y. There exists a X_j with $\neg[a]P \in X_j^-$ (possibly $\neg[a]P$ occurs as an n-formula; we ignore this now). W.l.o.g. none of the $X_i > X_j$ is obtained using $(M-)$ or $(M+)$.

↓ page 37

We now build the path $Y_j \lhd \ldots \lhd Y_n$. Let Y_j result from X_j^- by marking $\neg[a]P$ with P, and for $k > j$ let Y_k result from Y_{k-1} in the same way as X_k^- results from X_{k-1}^- (such that always $Y_k^- = X_k^-$).

There can be two cases:

(1) Some Y_k with $k > j$ is free. Then we have $Y_j \xrightarrow{a} Y_k$, and in particular $(S, S) \in \pi'(a)$ and $\neg P \in S$.

(2) All Y_k for $k > j$ are loaded. The node Y_n then contains a formula of the form $\neg[A][a_1] \ldots [a_l]P^P$. But now there exists a tableau $\mathcal{T}' \in M_0$ which has the root X_n^- which has the successor Y_n and to which $(M-)$ is not applied. By Lemma 12 Y_n has within this tableau a first free normal consistent successor Z, such that every normal node on the path from Y_n to Z is consistent. Moreover there exists in $\mathcal{T}'$ a simple consistent successor Z' of Z which is in the same local sub-tableau of $\mathcal{T}'$ as Z, because $\mathcal{T}'$ is maximal.

Now consider the path

$$X_1 \lhd \ldots \lhd X_j \lhd Y_j \lhd \ldots \lhd Y_n \lhd \ldots \lhd Z \lhd \ldots \lhd Z'$$

Every normal node of the path is consistent. Every maximal uncritical sub-path of this path ends with a simple node, because $\mathcal{T}'$ is saturating. For all nodes V, W which are roots and end nodes of an uncritical sub-path there exists a $S_{V,W} \in g$. In particular there exists an $S' \in g$ with $Z \subseteq S'$ and therefore $\neg P \in S'$. By Lemma 10 we have $Y_j \xrightarrow{a} Z$, and therefore $(S, S') \in \pi'(a)$.

It follows that $\mu = (g, \nu, \pi)$ also fulfils condition (iv). □

Theorem 4 (Completeness)**.** *Any set of formulas $X \subseteq \mathcal{F}$ is consistent if and only if it is satisfiable.*

Proof. From the Model Existence Theorem 3 and Lemma 9 it follows that any consistent set is satisfiable. Together with the Correctness Theorem 2 the claim follows. □

↓ page 38

1.11 Extended PDL-Tableaus

We can often extend a complete tableau-calculus by adding more rules, such that the construction of closed tableaus for inconsistent sets is simplified. We will make use of this in Part 2. There we will also use (in addition) the following rules:

$$(*;)\ \frac{X;[a^*;b]P}{X;[b]P \quad | \quad X;[a][a^*;b]P} \qquad (?;)\ \frac{X;[1?;a]P}{X;[a]P}$$

$$(F-)\ \frac{X;Y}{X}\ \ Y \text{ free} \qquad (1?)\ \frac{X;[1?]P}{X;P}$$

In addition, all applications of rules to normal free formulas may be interpreted as replacements (i.e. as done so far) or as addition of formulas. A node $X;P\wedge Q$ can thus get the successor $X;P\wedge Q;P;Q$ by application of $(\wedge)$.

Definition 23. *An* extended *PDL-tableau is a tableau constructed using the rules defined so far. Such a tableau is called closed if and only if all normal free end nodes are closed.*

One easily checks that for a satisfiable set X there is no closed extended PDL-tableau. It follows immediately that a set X is satisfiable if and only if there is no closed extended tableau. Providing a closed extended tableau for a set X thus also proves that it is not satisfiable.

↓ page 39

2 Interpolation in PDL

2.1 The Interpolation Theorem

Many modal logics, including those mentioned in Section 1.2, have an interpolation property. This can be proven using suitable complete tableau-calculi (RAUTENBERG [12]). In this part we will state an interpolation property for PDL and use the PDL-calculus to prove that PDL does in fact have this property.

From now on let $\mathcal{F}_0(P)$ and $\mathcal{P}_0(P)$ denote the sets of propositional variables and atomic programs, respectively, which appear in the formula P. Moreover, let $S(P) := \mathcal{F}_0(P) \cup \mathcal{P}_0(P)$. Similarly, let $S(a)$ be defined for all programs a. Finally, for any set of formulas X, let $S(X) := \cup_{P \in X} S(P)$.

Definition 24. *An interpolant for the pair (X_1, X_2) of sets of formulas is a formula R such that $S(R) \subseteq S(X_1) \cap S(X_2)$ and for which $X_1; \neg R$ and $X_2; R$ are inconsistent.*

Theorem 5 (Craig Interpolation)**.** *Let P, Q be formulas such that $\vDash P \to Q$. Then there is a formula R such that $S(R) \subseteq S(P) \cap S(Q)$ and for which we have $\vDash P \to R$ and $\vDash R \to Q$.*

Somewhat imprecisely we will also call a formula R with this property an interpolant for P and Q. By our definition it is actually an interpolant for the pair $(\{P\}, \{\neg Q\})$. In the following we will show that for every pair (X_1, X_2) such that $X_1; X_2$ is inconsistent there exists an interpolant. This implies the Craig Interpolation Theorem 5 with $X_1 = \{P\}$ and $X_2 = \{\neg Q\}$, for the inconsistency of $P; \neg Q$ means exactly $\vDash P \to Q$.

↓ page 40

2.2 Tableaus for Pairs of Sets of Formulas

We now want to introduce the concept of a tableau for a pair X_1/X_2 of sets of formulas. Such tableaus are again finite trees $(\mathcal{T}, \lhd)$; but each node $t \in \mathcal{T}$ is assigned a pair of sets of formulas $x_1(t)/x_2(t)$ by two functions x_1 and x_2. We often identify nodes with the pair which is assigned to it. A node Y_1/Y_2 is called free, loaded, closed, normal, n-node or $\neg[a^{(n)}]$-node, whenever $Y_1; Y_2$ is called like that. Such a tableau is constructed starting at the root X_1/X_2 by application of the rules of the PDL-calculus. We have to explain though, how rules should be applied to a formula in a component of the pair Y_1/Y_2.

- If the application of any rule (but not (At)) to a formula $P \in Y_1$ (where Y_1 is a node of a PDL-tableau) produces the formulas $Z_1, \ldots, Z_n$ ($n = 1$

or $n = 2$), then the application of the same rule to $P \in Y_1$ (Y_1 component of a node Y_1/Y_2) produces the pairs $Z_1/Y_2, \ldots, Z_n/Y_2$. Analogously for the application to a formula $P \in Y_2$.

- When (At) is applied to a formula $\neg[A]P^R \in Y_1$ (where Y_1 is a component of the node Y_1/Y_2 and $Y_1 = Y; \neg[A]P^R$ for some Y) we obtain the pair $Y_A; \neg P^{R\setminus P}/(Y_2)_A$. Analogously for the application to a formula $\neg[A]P^R \in Y_2$.

During this construction also the conditions 1 to 5 and 7 from Definitions 11 and 14 must be obeyed. The condition 6 however must be formulated differently for tableaus for pairs.

6'. A normal node $x_1(t)/x_2(t)$ for which there exists an $s < t$ such that $x_1(s) = x_1(t)$ and $x_2(s) = x_2(t)$, is an end node whenever the path from s to t is critical and when in addition the path is loaded in the case that $x_1(t)/x_2(t)$ is loaded.

↓ page 41

Additionally in the construction of a tableau for a pair the following condition must be obeyed:

8. If there are two nodes s and t with the same loaded components in a loaded sub-tableau, and if their respective successors result from an application of a rule (not $(M-)$) to the loaded components, then the same rule is applied to both nodes on the same formula.

Definition 25. *We call a tableau for a pair X_1/X_2 closed, if all normal free end nodes are closed. We call a normal pair X_1/X_2 inconsistent, if there exists a closed tableau for X_1/X_2, otherwise we call it consistent.*

Lemma 13. *A normal pair X_1/X_2 is consistent if and only if $X_1; X_2$ is consistent.*

Proof. If X_1/X_2 is inconsistent, then there is a closed tableau for X_1/X_2. By replacing each node Y_1/Y_2 by $Y_1; Y_2$ we essentially obtain an extended PDL-tableau. The only problem might be that some nodes $s < t$ violate condition 6 from Definition 14. If t is loaded, then one can simply "cut off" the tableau there; otherwise one can "cut out" the path from s to t to make an extended PDL-tableau, as defined in Section 1.11. This tableau is also closed, and thus $X_1; X_2$ is inconsistent.

For the other direction, if X_1/X_2 is consistent, then every tableau for X_1/X_2 is not closed. One can then (again by replacing each node Y_1/Y_2 by $Y_1; Y_2$) use these tableaus to construct a model graph as described in the Model

Existence Theorem 3 in which some state includes the set $X_1; X_2$. Hence in this case also $X_1; X_2$ is consistent. □

Corollary 1. *If $X_1; X_2$ is inconsistent, then there exists a closed tableau for X_1/X_2.*

↓ page 42

2.3 Construction of the Interpolant

In this section we explain how to construct an interpolant for X_1/X_2 when $X_1; X_2$ is inconsistent. For this we need a closed tableau $\mathcal{T}$ for X_1/X_2. This tableau will first be simplified.

Definition 26. *If $\mathcal{T}$ is a tableau for the pair X_1/X_2, then the tableau $\mathcal{T}^I$ is obtained from $\mathcal{T}$ by removing all n-nodes. Therefore, if $s < t$ are two normal nodes in $\mathcal{T}$ between which there are only n-nodes, then we have $s \lhd t$ in $\mathcal{T}^I$.*

We note that $x_1(s)/x_2(s)$ and $x_1(t)/x_2(t)$ differ in only one component. To see this, suppose the rule (n) or $(\neg n)$ is applied to a formula in $x_1(s)$, then (by condition 3) at every node u where $s < u < t$ a rule is applied to $x_1(u)$ (and not $x_2(u)$) because only this set contains an n-formula.

Because $\mathcal{T}^I$ does not contain any n-node, there is also no end node of $\mathcal{T}^I$ which would be a $\neg[a^{(n)}]$-node. Every end node is thus closed or has a predecessor with the same pair. If in fact $\mathcal{T}^I$ has only closed end nodes, then one can construct an interpolant for X_1/X_2 directly from $\mathcal{T}^I$. This is done by first defining an interpolant for each end node of $\mathcal{T}^I$.

In the following we will denote formulas that are interpolants for certain pairs with I, I_1, I_2,....

Lemma 14. *For every closed node Y_1/Y_2 of $\mathcal{T}^I$ there exists an interpolant.*

Proof. The pair Y_1/Y_2 is closed, if $Y_1; Y_2$ is closed. We thus have at least one of the following cases:

↓ page 43

- $0 \in Y_1$ or $P; \neg P \subseteq Y_1$ for some formula P:
  then we have $Y_1 \vDash 0$ and $Y_2 \vDash \neg 0$, hence let $I := 0$.
- $0 \in Y_2$ or $P; \neg P \subseteq Y_2$ for some formula P:
  then we have $Y_1 \vDash \neg 0$ and $Y_2 \vDash 0$, hence let $I := \neg 0$.
- $P \in Y_1$ and $\neg P \in Y_2$ for some formula P:
  then we have $Y_1 \vDash P$ and $Y_2 \vDash \neg P$, hence let $I := P$.
- $\neg P \in Y_1$ and $P \in Y_2$ for some formula P:
  then we have $Y_1 \vDash \neg P$ and $Y_2 \vDash P$, hence let $I := \neg P$.

Moreover, in all cases we have $S(I) \subseteq S(Y_1) \cap S(Y_2)$, hence in all cases I is an interpolant of Y_1/Y_2. □

We can then define interpolants of each node for which the successors already have interpolants. Intuitively, we construct the interpolant of X_1/X_2 "from the leaves to the root".

Lemma 15. *Let Y_1/Y_2 be a node of $\mathcal{T}^I$ with n direct successors for which interpolants $I_1, \ldots, I_n$ have already been found. Then there exists an interpolant for Y_1/Y_2.*

Proof. If the n successors are obtained using any rule besides (At), then we have one of the following cases:

- The successors differ from Y_1/Y_2 only in the second component, they are of the form $Y_1/Z_1, \ldots, Y_1/Z_n$. Obviously we have $Y_1 \vDash I_1 \wedge \ldots \wedge I_n$. Moreover Y_2 only holds in a state in which also one of the Z_i holds. But then from $Z_i \vDash \neg I_i$ for each $i \leq n$ we get that $Y_2 \vDash \neg I_1 \vee \ldots \vee \neg I_n$, and thus $Y_2 \vDash \neg(I_1 \wedge \ldots \wedge I_n)$. Hence let $I := I_1 \wedge \ldots \wedge I_n$. Because we have $S(Z_i) \subseteq S(Y_2)$ for each i, it follows that $S(I) \subseteq S(Y_1) \cap S(Y_2)$; hence I is an interpolant for Y_1/Y_2.
- The successors differ from Y_1/Y_2 only in the first component, they are of the form $Z_1/Y_2, \ldots, Z_n/Y_2$. Obviously we have $Y_2 \vDash \neg I_1 \wedge \ldots \wedge \neg I_n$ and thus $Y_1 \vDash \neg(I_1 \vee \ldots \vee I_n)$. Moreover Y_1 only holds in a state in which also one of the Z_i holds. But then from $Z_i \vDash I_i$ for each $i \leq n$ we get that $Y_1 \vDash I_1 \vee \ldots \vee I_n$. Hence let $I := I_1 \vee \ldots \vee I_n$. Because we have $S(Z_i) \subseteq S(Y_2)$ for each i, it follows that $S(I) \subseteq S(Y_1) \cap S(Y_2)$; hence I is an interpolant for Y_1/Y_2.

↓ page 44

Before we consider the second case (i.e. the (At) rule), we note that $Y_A \vDash P$ always implies $Y \vDash [A]P$. To see this, suppose $S \Vdash Y$ for some (g, ν, π) and $S \in g$. Then we have $T \vDash Y_A$ and thus $T \Vdash P$ for all $T \in g$ with $S \xrightarrow{A} T$. But this means $S \Vdash [A]P$.

Moreover $Y_A; \neg R \vDash P$ always implies $Y; \neg[A]R \vDash \neg[A]\neg P$, because if we have $S \Vdash Y; \neg[A]R$ for some (g, ν, π) and $S \in g$, then there exists a $T \in g$ with $S \xrightarrow{A} T$ and $T \Vdash Y_A; \neg R$. Because then $T \Vdash P$, we have $S \Vdash \neg[A]\neg P$.

If now Y_1/Y_2 has a successor Z_1/Z_2 which is obtained using the critical rule (At), and for which there already is an interpolant I, then we have one of the following cases:

- Y_1 contains a marked formula of the form $\neg[A]P^R$. Then we have $Z_1 = \neg P^{R\setminus P}; (Y_1)_A$ and $Z_2 = (Y_2)_A$.

  If Z_2 is non-empty, then $A \in S(Y_1) \cap S(Y_2)$. As we have seen, $Y_1 \vDash \neg[A]\neg I$ follows from $\neg P^{R\setminus P}; (Y_1)_A \vDash I$, and $Y_2 \vDash [A]\neg I$ follows from $(Y_2)_A \vDash \neg I$. Hence $\neg[A]\neg I$ is an interpolant for Y_1/Y_2

  If Z_2 is empty, then Y_1 is already inconsistent. Then we have $Y_1 \vDash 0$ and $Y_2 \vDash \neg 0$, and 0 is an interpolant for Y_1/Y_2.

- Y_2 contains a marked formula of the form $\neg[A]P^R$. Then we have $Z_1 = (Y_1)_A$ and $Z_2 = \neg P^{R\setminus P}; (Y_2)_A$.

  If Z_1 is non-empty, then $A \in S(Y_1) \cap S(Y_2)$. As we have seen, $Y_1 \vDash I$ follows from $(Y_1)_A \vDash I$ and $Y_2 \vDash \neg[A]I$ follows from $\neg P^{R\setminus P}; (Y_2)_A \vDash I$.

  Hence $[A]I$ is an interpolant for Y_1/Y_2

  If Z_1 is empty, then Y_2 is already inconsistent. Then we have $Y_1 \vDash \neg 0$ and $Y_2 \vDash 0$, and $\neg 0$ is an interpolant for Y_1/Y_2.

This shows the lemma. □

However, in general $\mathcal{T}^I$ will not only have closed end nodes. Recall that also those loaded nodes are end nodes which have a predecessor in the same loaded sub-tableau of $\mathcal{T}^I$ with the same pair.

↓ page 45

Hence in general it is not possible to construct an interpolant for the root of a loaded sub-tableau using the methods described in Lemmas 14 and 15, because there might be successors of this root that are not closed end nodes. In the following we explain, how one can still construct interpolants for such roots, as long as interpolants are known for their first free successors.

Let thus Y_1/Y_2 be a node of $\mathcal{T}^I$ which was obtained using $(M+)$; then Y_1/Y_2 is the root of a loaded sub-tableau of $\mathcal{T}^I$. Suppose that for all free successors of Y_1/Y_2 we already have interpolants. We can then restrict our attention to the case where Y_2 contains the marked formula. For if that is not the case, then we can swap the first and second component in all nodes of the tableau (if I is an interpolant for Y_1/Y_2, then $\neg I$ is an interpolant for Y_2/Y_1; the interpolants found so far are thus to be replaced by their negations). Moreover, we can assume $Y_1 \neq \varnothing$, because if we have $Y_1 = \varnothing$, then Y_2 is already inconsistent and $\neg 0$ is an interpolant for Y_1 and Y_2.

We will now define, which parts of $\mathcal{T}^I$ we need for the construction of the interpolant.

Definition 27. *Let $\mathcal{T}^J$ be a sub-tableau of $\mathcal{T}^I$ with the root Y_1/Y_2. Let a node $Z_1/Z_2 > Y_1/Y_2$ be an end node of $\mathcal{T}^J$ if it is minimal such that it fulfils one of the following conditions:*

- *At Z_1/Z_2 the rule $(M-)$ is applied.*
- *Z_1/Z_2 is free.*
- *The first component of its successor is empty.*
- *There is a predecessor of Z_1/Z_2 with the same pair.*

↓ page 46

If a node has a predecessor to which the same pair is assigned, then we will also call that other node a *predecessor with the same pair* in the following.

Lemma 16. *$\mathcal{T}^J$ has the following properties:*

(a) Every component of every node is nonempty.

(b) For each end node t that has no predecessors with the same pair an interpolant $I(t)$ is known.

(c) There is at least one end node for which an interpolant is known.

Proof. It is obvious that (a) holds.

Let t be an end node of $\mathcal{T}^J$ which has no predecessor with the same pair. If $x_2(t)$ is free, then an interpolant for t is already known. If $x_2(t)$ is not free, then $(M-)$ is applied to t, or the first component of its successor t' is empty (which can only happen if (At) is applied to t). In the first case t' is free, whereby an interpolant for t' and thus also for t is already known. In the second case though already $x_2(t)$ is inconsistent, and $\neg 0$ is thus an interpolant for t. Hence we have (b).

In $\mathcal{T}^I$ the node Y_1/Y_2 has at least one free successor t. For this node an interpolant is known. Either this node is also in $\mathcal{T}^J$, or there is a node $s < t$ which is an end node of $\mathcal{T}^J$. But for s there is no successor with the same pair in $\mathcal{T}^J$. Hence by (b) an interpolant for s is known. Therefore we also have (c). □

Definition 28. *For each node t of $\mathcal{T}^J$ let $K(t)$ be the conjunction (Konjunktion) over all $P \in x_1(t)$. For any set T of nodes of $\mathcal{T}^J$ let $D(T)$ be the disjunction over all $K(t)$ with $t \in T$. For $T = \varnothing$ let $D(T) := 0$. We note that for any $T_1 \subseteq T_2$ we have that $D(T_2) \vDash P$ implies $D(T_1) \vDash P$. Moreover, let $T(Y)$ denote the set of all nodes t of $\mathcal{T}^J$ with $x_2(t) = Y$.*

↓ page 47

For each Y appearing as a second component of a node in $\mathcal{T}^J$ we define the following partition of the set $T(Y)$:

Definition 29. *The set $T(Y)^\epsilon$ consists of all nodes $s \in T(Y)$ for which there is no node $t \in T(Y)$ with $s \lhd' t$. Furthermore, let $T(Y)^I$ be the set of those nodes $s \in T(Y)^\epsilon$ for which there exists no node $t \in \mathcal{T}^J$ with $s \lhd' t$. Finally, let $T(Y)^\lhd$ be the set $T(Y)^\epsilon \setminus T(Y)^I$.*

Each node $s \in T(Y)^I$ is an end node of $\mathcal{T}^J$ for which an interpolant $I(t)$ is known, because it has no predecessor with the same pair. To any node $t \in T(Y)^\lhd$ that is not an end node of $\mathcal{T}^J$ a rule is applied to $x_2(t)$. If Y is free, then $T(Y)$ only consists of end nodes, and we have $T(Y)^\lhd = \varnothing$ and $T(Y)^\epsilon = T(Y)^I$.

Lemma 17. *From $D(T(Y)^\epsilon) \vDash P$ it follows that $D(T(Y)) \vDash P$.*

Proof. The set $T(Y)$ together with the relation $\lhd'$ is essentially a set of trees, of which the end nodes are exactly the nodes of $T(Y)^\epsilon$. For the nodes $s, t_1, \ldots, t_n \in T(Y)$, where the t_i are the $\lhd'$-successors of s, we have that $K(t_i) \vDash P$ for all i always implies $K(s) \vDash P$. Hence we have $K(s) \vDash P$ even for all $s \in T(Y)$. □

Definition 30. *If $T(Y)^I = \{t_1, \ldots, t_n\}$ with $n > 0$, then let $I(Y)$ be the disjunction over all interpolants $I(t_i)$, if $n = 0$ let $I(Y) := 0$.*

Lemma 18. *If $T(Y)^\lhd = \varnothing$, then $I(Y)$ is an interpolant for $D(T(Y))/Y$.*

Proof. Let $T(Y)^I = \{t_1, \ldots t_n\}$. Because $D(T(Y)^I)$ contains exactly the $K(t_i)$ as disjuncts, and because we always have $K(t_i) \vDash I(t_i)$, we also have $D(T(Y)^I) \vDash I(Y)$. Because $T(Y)^I = T(Y)^\epsilon$, by Lemma 17 we have $D(T(Y)) \vDash I(Y)$. Moreover, because $Y \vDash \neg I(t_i)$ holds for all $t_i \in T(Y)^I$, we also have $Y \vDash \neg I(Y)$. In addition we have $S(I(t_i)) \subseteq S(x_1(t_i)) \cap S(Y)$ for all i, and thus the claim holds. □

↓ page 48

Lemma 19. *If $D(T(Y)^\lhd) \vDash P$, then $D(T(Y)) \vDash [\neg I(Y)?]P$.*

Proof. If $t \in T(Y)^I$, then $K(t) \vDash I(Y)$. If $t \in T(Y)^\lhd$, then $K(t) \vDash P$.

Hence, if $t \in T(Y)^\epsilon = T(Y)^I \cup T(Y)^\lhd$, then $K(t) \vDash I(Y) \vee P$, and thus also $K(t) \vDash [\neg I(Y)?]P$. Hence $D(T(Y)^\epsilon) \vDash [\neg I(Y)?]P$ and by Lemma 17 the claim follows. □

We now define a tableau where to each node t we assign a pair $D(T(Y))/Y$ or $D(T(Y)^\lhd)/Y$ together with a number $k(t) \in \{1, 2, 3\}$. We denote such a

node with $D(T(Y))/_{k(t)}Y$ or $D(T(Y)^{\lhd})/_{k(t)}Y$. Here Y is always the second component of a node of $\mathcal{T}^J$.

Definition 31. *The tableau $\mathcal{T}^K$ has as its root the node $D(T(Y_2))/_1Y_2$. A node $D(T(Y))/_1Y$ of the tableau, for which no predecessor t with the same pair and $k(t) = 1$ exists, has exactly one successor $D(T(Y))/_2Y$, otherwise it is an end node. A node $D(T(Y))/_2Y$ has a successor, namely $D(T(Y)^{\lhd})/_3Y$ iff $T(Y)^{\lhd}$ is nonempty; otherwise it is an end node. We note that $T(Y)^{\lhd}$ is empty when Y is free. If $T(Y)^{\lhd}$ is nonempty, then the same rule is applied to every node $t \in T(Y)^{\lhd}$ in $\mathcal{T}^J$ to the same formula. Hence there are sets of formulas $Z_1, \ldots, Z_n$ such that each node $t \in T(Y)^{\lhd}$ has exactly n successors with the second components $Z_1, \ldots, Z_n$. Then let the successors of $D(T(Y)^{\lhd})/_3Y$ be exactly the pairs $D(T(Z_1))/_1Z_1, \ldots, D(T(Z_n))/_1Z_n$.*

If K is the number of different second components of $\mathcal{T}^J$, then $\mathcal{T}^K$ is not longer than $3K + 1$.

In addition we define for all $s, t \in \mathcal{T}^K$ with $s \lhd t$ a program $\mathcal{P}(s, t)$ which we call the *canonical program* from s to t. If then s, t are any nodes with $s < t$ and is $s = s_1 \lhd \ldots \lhd s_n = t$ the path from s to t, and is $a_i = \mathcal{P}(s_i, d_{i+1})$ where $i < n$ the canonical program from s_i to s_{i+1}, then let $\mathcal{P}(s, t) := a_1; \ldots; a_{n-1}$ be the canonical program from s to t. For $a = \mathcal{P}(s, t)$ we also write $s \xrightarrow{a}{}' t$.

↓ page 49

In the following by an 'i-node s' we mean a node s of $\mathcal{T}^K$ with $k(s) = i$. Moreover with $s^i, t^i, \ldots$ we always denote arbitrary i-nodes..

Definition 32. *Let s^j be a node such that for all t, t' with $s^j \leq t \lhd t'$ the canonical program from t to t' is already defined and let $s^i \lhd s^j$.*

- *If $i = 2$ and $j = 3$, then s^i is assigned a pair $D(T(Y))/_2Y$ and s^j a pair $D(T(Y)^{\lhd})/_3Y$. Let $\mathcal{P}(s^i, s^j) := \neg I(Y)?$.*
- *If $i = 1$ and $j = 2$, then s^i is assigned a pair $D(T(Y))/_1Y$ and s^j a pair $D(T(Y))/_2Y$. Let $t_1, \ldots, t_n$ be all the successors of s^i with the same pair and $k(t_l) = 1$ for all $1 \leq l \leq n$. If $n = 0$, then let $\mathcal{P}(s^i, s^j) := 1?$. If $n > 0$, then let a_l be the canonical program from s^j to t_l. In this case, let $\mathcal{P}(s^i, s^j) := (a_1 \cup \ldots \cup a_n)^*$.*
- *If $i = 3$ and $j = 1$, then s^i is assigned a pair $D(T(Y)^{\lhd})/_3Y$ and s^j a pair $D(T(Z)^{\lhd})/_1Z$. The set Z here is obtained by applying some rule. If this rule is not (At), then let $\mathcal{P}(s^i, s^j) := 1?$. Otherwise we have $Y = Y'; \neg[A]P^R$ and $Z = Y'_A; \neg P^{R \setminus P}$ for some Y' and $\neg[A]P^R$. In this case let $\mathcal{P}(s^i, s^j) := A$.*

It follows from this definition that if we have $a = \mathcal{P}(s,t)$ and if $D(T)/_iY$ is the pair assigned to the node s, then we have $S(a) \subseteq S(D(T)) \cap S(Y)$.

Lemma 20. *Let $s^i < s^j$. Let s^i be assigned the pair $D(T)/_iY$ and s^j the pair $D(T')/_jY$. If a is the canonical program from s^i to s^j, then $D(T') \vDash P$ implies $D(T) \vDash [a]P$.*

Proof. First, let $s^i \lhd s^j$ and suppose the claim is already proven for the sub-tableau beginning at s^j (induction hypothesis).

- If $i = 2$ and $j = 3$, then $Y = Y'$, $T = T(Y)$ and $T' = T(Y)^{\lhd}$, and $a = \neg I(Y)?$. The claim follows by Lemma 19.
- If $i = 1$ and $j = 2$, then $Y = Y'$ and $T = T' = T(Y)$. If $a = 1?$, then the claim is trivial. Suppose $a = (a_1 \cup \ldots a_n)^*$. Then there are successors $t_1, \ldots, t_n$ of s^j, which have the pair $D(T)/Y$. For each t_k where $k \leq n$ let a_k be the canonical program from s^j to t_k. From the induction hypothesis it follows that $D(T) \vDash [a_k]P$ for all k, hence we also have $D(T) \vDash [a_1 \cup \ldots \cup a_n]P$. But if $D(T) \vDash [a_1 \cup \ldots \cup a_n]P$ follows from $D(T) \vDash P$, then it also follows that $D(T) \vDash [(a_1 \cup \ldots \cup a_n)^*]P$.
- If$i = 3$ and $j = 1$, then $T = T(Y)^{\lhd}$ and $T' = T(Y')$. If $a = 1?$, then for all $t \in T$ the conjunction $K(t)$ is a disjunct of $D(T')$ and we have $K(t) \vDash P$. It follows that $D(T) \vDash P$ and $D(T) \vDash [1?]P$. If $a = A$ for an atomic program A, then for all $t \in T$ the conjunction over $x_1(t)_A$ is a disjunct of $D(T')$, and we have $x_1(t) \vDash [A]P$. It follows that $D(T) \vDash [A]P$.

↓ page 50

If now $s^i \not\lhd s^j$ and $s_1; \ldots; a_n = \mathcal{P}(s^i, s^j)$, then $D(T') \vDash P$ already implies $D(T) \vDash [a_1] \ldots [a_n]P$, which means $D(T) \vDash [a_1; \ldots; a_n]P$. □

Lemma 21. *For each end node of $\mathcal{T}^K$ that does not have a predecessor with the same pair, there is an interpolant. Moreover, $\mathcal{T}^K$ has at least one such node.*

Proof. Each end node $\mathcal{T}^K$ that does not have a predecessor with the same pair is assigned a pair $D(T(Y))/_2Y$ for some Y such that $T(Y)^{\lhd}$ is empty. By Lemma 18 we have that $I(Y)$ is an interpolant for this pair.

The tableau obtained from $\mathcal{T}^K$ by removing all first components of all pairs and "taking out" all 2-nodes and 3-nodes, is a sub-tableau of a PDL-tableau with the root Y_2. (Is this a typo? What is Y_2 here?)This tableau has at least one end node Y that does not have a predecessor with the same pair. Hence $D(T(Y))/_2Y$ is the end node of $\mathcal{T}^K$ which we were looking for. □

↓ page 51 **Definition 33.** *Let t be an end node of $\mathcal{T}^K$, and let t be assigned the pair $D(T)/Y$. If t has no predecessor with the same pair, let $I(t) := I(Y)$. For all other end nodes let $I(t) := 1$.*

Let s be a non-end node of $\mathcal{T}^K$ and let $s_1, \ldots, s_n$, $n \geq 1$, be the direct successors of s. If $n = 1$, $a = \mathcal{P}(s, s_1)$ and $a \neq 1?$, then let $I(s) := [a]I(s_1)$. Otherwise let $I(s) := I(s_1) \wedge \ldots \wedge I(s_n)$. If t_0 is the root of $\mathcal{T}^K$, then let $I_0 := I(t_0)$.

We now show that the formula I_0 constructed in this way is an interpolant for Y_1/Y_2.

Lemma 22. *We have $S(I_0) \subseteq S(Y_1) \cap S(Y_2)$.*

Proof. First we have $S(D(T)) \subseteq S(Y_1)$ and $S(Y) \subseteq S(Y_2)$ for each pair $D(T)/Y$ in $\mathcal{T}^K$. Hence for every end node t we already have $S(I(t)) \subseteq S(Y_1) \cap S(Y_2)$ and, by the remark right after Definition 32 above, also $S(a) \subseteq S(Y_1) \cap S(Y_2)$. Hence the claim follows. □

Lemma 23. *We have $Y_1 \vDash I_0$.*

Proof. We now distinguish the same cases as in Definition 33.

For each end node t of $\mathcal{T}^K$ we have $x_1(t) \vDash I(t)$ (either trivially, if $I(t) = 1$, or by Lemma 16).

Let $s_1, \ldots, s_n$ be the direct successors of s, and suppose we have $x_1(s_i) \vDash I(s_i)$ for $1 \leq i \leq n$. If $n = 1$, $a = \mathcal{P}(s, s_1)$ and $I(s) = [a]I(s_1)$, then $x_1(s) \vDash I(s)$ is exactly the claim of Lemma 20. If $n \geq 1$ and $1? = \mathcal{P}(s, s_i)$ for all i, then (still using Lemma 20) from $x_1(s) \vDash [1?]I(s_i)$ for all i it follows that $x_1(s) \vDash I(s_1) \wedge \ldots \wedge I(s_n)$, hence $x_1(s) \vDash I(s)$. Therefore by induction we also have $x_1(t_0) \vDash I_0$. Notably, the conjunction over Y_1 is a disjunct of $x_1(t_0) = D(T(Y_2))$, hence we also have $Y_1 \vDash I_0$. □

Definition 34. *Let $J(t_0) := \varnothing$. Suppose for all $t < s$ we already defined $J(t)$. Then let $J(s)$ be the smallest set such that for all nodes t, t' of $\mathcal{T}^K$ with the same pair and $t < s \leq t'$, and for all formulas $P \in J(t) \cup \{I(t)\}$ it holds that $[\mathcal{P}(s, t')]P \in J(s)$ (for $s = t'$: that $P \in J(s)$). Moreover, let $K(s) := \{I(s)\} \cup J(s) \cup x_2(s)$.*

↓ page 52 **Lemma 24.** *We have:*

(a) $K(t_0) = I_0; Y_2$

(b) *If t is an end node of $\mathcal{T}^K$ which has no predecessor with the same pair, then $K(t)$ is inconsistent.*

(c) *If $t < t'$ are nodes with the same pair, then $K(t) \subseteq K(t')$.*

Proof. All three claims follows easily from the definitions. □

Lemma 25. *If $s_1, \ldots, s_n$ are the direct successors of the node $s^i \in \mathcal{T}^K$, then there is an extended tableau for $K(s^i)$ with the end nodes $K(s_1), \ldots, K(s_n)$ as well as (possibly) further inconsistent free end nodes.*

Proof. We consider five cases. In the first three cases we have $i = 1$ or $i = 2$, and s^i has exactly one successor s^{i+1}. Here we always have $x_2(s^i) = x_2(s^{i+1})$, hence $J(s^{i+1})$ and $I(s^{i+1})$ should be constructed from $K(s^i)$.

Case 1 We have $i = 1$, and there are m many 1-nodes $t_1, \ldots, t_m > s^1$ with the same pair as s^1. Then we have $s^1 \overset{b^*\,\prime}{\to} s^2$, where $b = a_1 \cup \ldots \cup a_m$ and $a_j = \mathcal{P}(s^2, t_j)$ for all $1 \leq j \leq m$. We have $I(s^1) = [b^*]I(s^2)$ and the application of $(*;)$ yields $I(s^2)$ together with $[b][b^*]I(s^2)$. Let furthermore $P \in J(s^2)$. There are 1-nodes u, u' in $\mathcal{T}^K$ with the same pair, with $u < s^2 < u'$ and the canonical program $s^2 \overset{a}{\to} u'$ such that $P = [a]Q$ for some $Q \in J(u) \cup \{I(u)\}$. Either $u < s^1$, then we already have $[b^*; a]Q \in J(s^1)$ and P can be constructed from this formula using $(*;)$, or we have $u = s^1$, $u' = t_j$ and $a = a_j$ for some $1 \leq j \leq m$. If $Q \in J(s^1)$, then $Q = [b^*; a_j]R$ for some R. Using $(*;)$ we get $[b][b^*; a_j]R$, and using $(\cup)$ j many times we get $P = [a_j]Q = [a_j][b^*; a_j]R$. If $Q = I(s^1)$, then we obtain already by j many applications of $(\cup)$ to the already obtained formula $[b][b^*]I(s^2)$ the formula $P = [a_j]Q = [a_j][b^*]I(s^2)$.

↓ page 53

Case 2 We have $i = 2$, and there are no 1-nodes $s > s^1$ with the same pair. Then we have $s^1 \overset{1?\,\prime}{\to} s^2$. If $P \in J(s^2)$, then $P = [a]Q$ for some a, Q and $[1?; a]Q \in J(s^1)$. By application of $(?;)$ one obtains $P = [a]Q$. Then by application of $(1?)$ we get $I(s^2)$ from $I(s^1) = [1?]I(s^2)$.

Case 3 We have $i = 2$, and with $Y = x_2(s^3)$ we have $s^2 \overset{\neg I(Y)?}{\to} s^3$. Suppose $P \in J(s^3)$. We have $P = [a]Q$ for some a, Q, and we have $[\neg I(Y)?; a]Q \in J(s^2)$. The application of $(;)$ yields $[\neg I(Y)?][a]Q$ and the application of $(?)$ yields two successors. One successor contains $\neg\neg I(Y); Y$ and is inconsistent. To this one $(M-)$ is applied, yielding a free inconsistent node. The other successor contains $P = [a]Q$. In the same way one builds $I(s^3)$ from $I(s^2) = [\neg I(Y)?]I(s^3)$.

For the two remaining cases we have $i = 3$. Then s^i has n many successors $u_1, \ldots, u_n$ where the $x_2(u_j)$ are obtained from $x_2(s^3)$ by application of a rule.

Case 4 We have $i = 3$, $n = 1$, and $x_2(u_1)$ is obtained from $x_s(s^3)$ by application of (At). Then we have $s^3 \stackrel{A}{\to} u_1$ for some atomic program. If $P \in J(u_1)$, then $[A]P \in J(s^3)$, or we have $P = [a]Q$ for some a, Q and $[A;a]Q \in J(s^3)$. In this case by application of $(;)$ one obtains the formula $[A]P = [A][a]Q$. Moreover, $I(s^3) = [A]I(u_1)$. But after this the application of (At) yields all formulas of $K(u_1)$.

Case 5 We have $i = 3$, $n \geq 1$, and the $x_2(u_j)$ are obtained by application of a rule other than (At) to $x_2(s^3)$. Moreover we have $s^3 \stackrel{1?}{\to}' u_j$ for all $j \leq n$, and if $P \in J(u_j)$, then $[1?]P \in J(s^3)$, or we have $P = [a]Q$ for some a, Q and $[1?;a]Q \in J(s^3)$. In any case one obtains P by application of $(1?)$ or $(?;)$. Finally, $I(s^3) = I(u_1) \wedge \ldots \wedge I(u_n)$. By $(n-1)$ many applications of $(\wedge)$ we obtain all $I(u_j)$.

In all cases we thus obtain nodes $K'_1, \ldots, K'_n$ with $K(s_j) \subseteq K'_j$ for all $1 \leq j \leq n$. By application of $(F-)$ to these nodes one finally obtains the end nodes $K(s_1), \ldots, K(s_n)$. □

↓ page 54

Lemma 26. *We have* $Y_2 \vDash \neg I_0$.

Proof. We construct an extended tableau for $I_0; Y_2^-$, by first applying $(M+)$ to obtain the node $K(t_0) = I_0; Y_2$. Starting from this node we use the method described in Lemma 25 to construct for all nodes $t \in \mathcal{T}^K$ the nodes $K(t)$. In this way we eventually obtain a tableau which as end nodes (besides possibly some free inconsistent nodes) has exactly the nodes $K(t_1), \ldots, K(t_n)$, where $t_1, \ldots, t_n$ are the end nodes of $\mathcal{T}^K$.

For each node t_i that has no predecessor with the same pair, $K(t_i)$ is inconsistent, and if it is not already free itself, by application of $(M-)$ we obtain a free inconsistent node.

For each node t_i that has a predecessor t'_i in $\mathcal{T}^K$ with the same pair, we have $K(t'_i) \subseteq K(t_i)$. By application of $(F-)$ to $K(t_i)$ we thus obtain a node for which there is a predecessor with the same set.

Hence we obtained an extended tableau for $I_0; Y_2^-$ where this set has only inconsistent first free successors. Thereby $I_0; Y_2^-$ is inconsistent, and we have $Y_2 \vDash I_0$. □

Lemma 27. *If* $X_1; X_2$ *is inconsistent, then there is an interpolant for* X_1/X_2.

Proof. First there exists a closed tableau for X_1/X_2. For each free normal end node of this tableau we can define an interpolant. For each node Y_1/Y_2 of the tableau, if interpolants are known for all its free successors, then using the method described in this section we can construct an interpolant. Hence step by step we can construct interpolants for every free node of the tableau, in particular for X_1/X_2. □

With this Lemma the proof of the Interpolation Theorem 5 is done.

↓ page 55

2.4 Practical Execution of a Construction

To construct an interpolant in practice, it is not necessary not go through all steps described in the previous section. For example, in the construction of the tableau $\mathcal{T}^K$ we do not have to determine the first components of each node. We only introduced them to prove the desired properties of the constructed interpolants.

To demonstrate which steps actually have to be taken, as an example we now construct an interpolant for the pair X_1/X_2 with

$$X_1 = \{[(A;A)^*](p \wedge [A;(B \cup C)]0)\} \quad \text{and} \quad X_2 = \{\neg[A^*](p \vee [C]q)\}.$$

We note that $S(X_1) = \{p, A, B, C\}$, $S(X_2) = \{p, q, A, C\}$ and $S(X_1) \cap S(X_2) = \{p, A, C\}$.

Step 1: Construction of a closed tableau $\mathcal{T}$ for X_1/X_2.
On the following page in Figure 1 we provide such a tableau $\mathcal{T}$. We use the abbreviations $Q = p \wedge [A;(B \cup C)]0$ and $P := p \vee [C]q$. Moreover, if a component of a node is identical to the same component of its predecessor, then we indicate this with a dot and do not write it down again.

Step 2: Construction of interpolants for all nodes which do not have a loaded successor that is an end node.
This can be done using Lemmas 15 and 16 without other intermediate steps. We have written these interpolants next to the corresponding nodes of $\mathcal{T}$.

↓ page 56

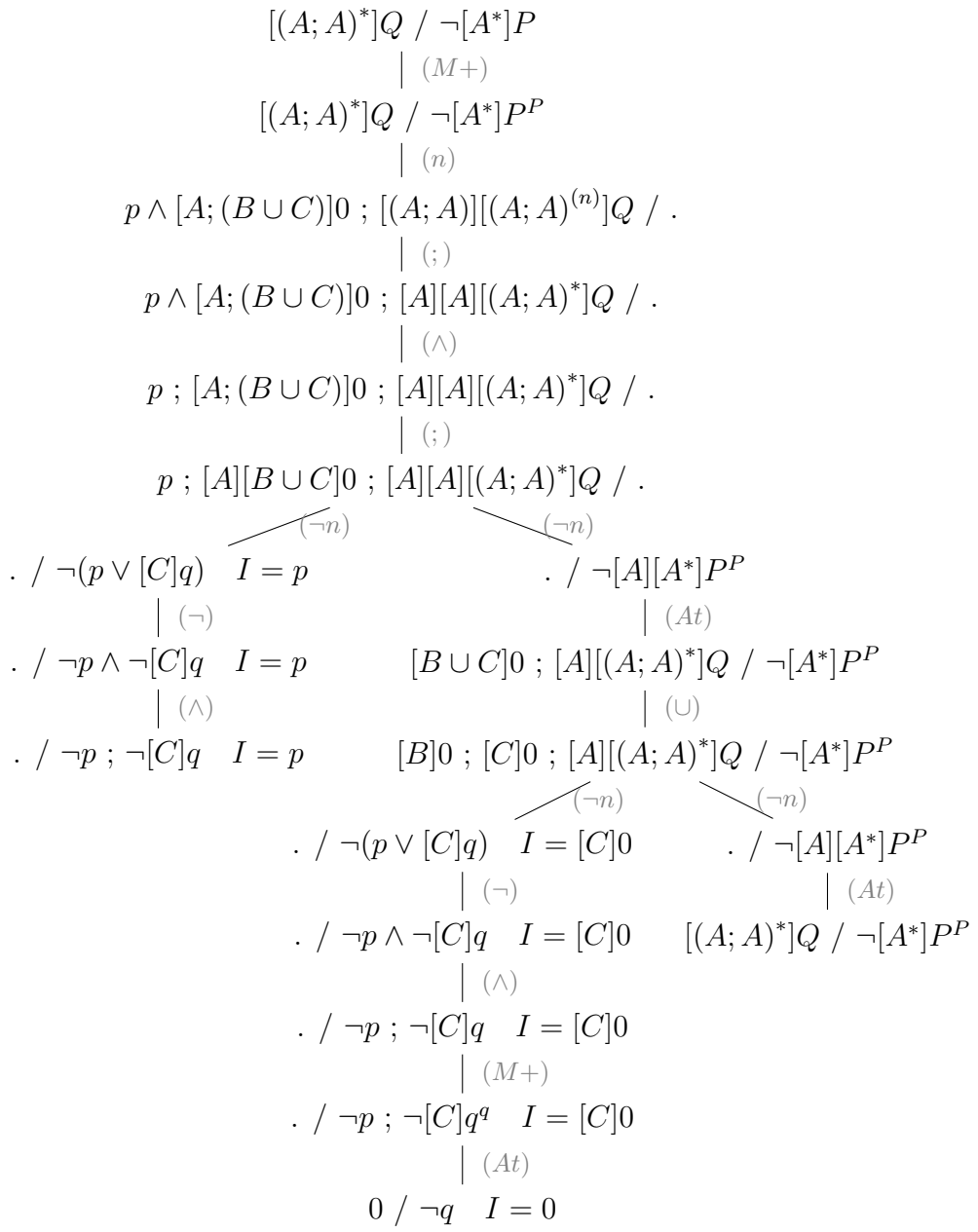


Figure 1: A closed tableau for X_1/X_2.

↓ page 57

There is now exactly one free node in the tableau, namely the root X_1/X_2 for which no interpolant is known. The remaining steps thus have the goal to build an interpolant for the node X_1/X_2 from those of its first free successors.

Step 3: Construction of $\mathcal{T}^I$ and $\mathcal{T}^J$.
For this we only need to decide which successors of X_1/X_2 belong to $\mathcal{T}^J$. These are all normal nodes where the second component is one of the following sets:

$$\begin{array}{lcl} Y_1 & := & \{\neg[A^*]P^P\} \\ Y_2 & := & \{\neg[A][A^*]P^P\} \\ Y_3 & := & \{\neg P\} \end{array}$$

Step 4: Determining the necessary $I(Y)$.
All nodes of $\mathcal{T}^J$ with the second component Y_3 are end nodes of $\mathcal{T}^J$, and for those we already have interpolants (p and $[C]0$). We thus have $T(Y_3)^{\triangleleft} = \varnothing$ and $I(Y_3) = p \vee [C]0$.

Step 5: Construction of $\mathcal{T}^K$ and I_0.
As already mentioned it is completely unnecessary to explicitly determine the formulas $D(T(Y_1))$ and so on. The form of the tableau $\mathcal{T}^K$ depends only on the second components of the nodes of $\mathcal{T}^J$. To construct the tableau, which consists of the second components of $\mathcal{T}^K$, it suffices to determine that Y_1 is its root and has the successors Y_2 and Y_3, that Y_1 is obtained by applying (At) to $\neg[A][A^*]P^P \in Y_2$ and that Y_3 is an end node.

One will see that $1?$ occurs often as a canonical program in $\mathcal{T}^K$. To get an interpolant that is not unnecessarily large, during the construction one may replace each occurring program $a;1?$ and $1?;a$ by the program a and each formula $[\neg P?]P$, $[1?]P$, $1 \wedge P$ or $P \wedge 1$ by the formula P.

↓ page 58

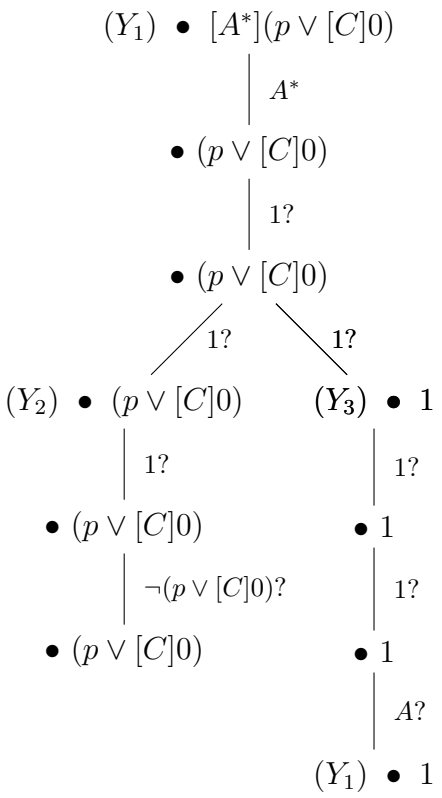


Figure 2: Untitled.

Result of the construction is thus, that $I_0 = [A^*](p \vee [C]0)$ is an interpolant for the formulas $[(A;A)^*](p \wedge [A;(B \cup C)]0)$ and $[A^*](p \vee [C]q)$. Due to the aforementioned consideration not to construct an unnecessarily large interpolant, we even have that I_0 is test-free. However, this cannot always be ensured.

Lemma 28. *There are formulas P and Q with $\vDash P \to Q$ such that no interpolant for P and Q is test-free.*

Proof. In BERMAN, PATERSON [2] it is shown that no test-free formula R is equivalent to the formula $P := \neg[(p?;A)^*;\neg p?;A;p?]0$, i.e. for every test-free formula R there is a Kripke model (g,ν,π) and an $S \in g$ with $S \Vdash P \iff S \Vdash \neg R$. Thereby we cannot have both $\vDash P \to R$ and $\vDash R \to P$. But because we have $\vDash P \to P$, this means that no test-free formula is an interpolant for P and P. □

↓ page 59

2.5 Open Questions

The result of Lemma 28 immediately motivates the following question:

- Is there for all test-free formulas P and Q with $\vDash P \to Q$ a test-free interpolant?

Note: Marcus Kracht shows in *Tools and Techniques in Modal Mogic* that "if test–free PDL has interpolation, then full PDL also has interpolation" (Theorem 10.6.2, page 495). This does *not* imply a positive answer to the above question, as was wrongly claimed here in this translation before.

This question could be answered positively if it was shown that for all test-free formulas P and Q with $\vDash P \to Q$ there exists a closed tableau $\mathcal{T}$ for $P/\neg Q$ such that for all nodes s^2 and s^3 in $\mathcal{T}^K$ with $s^2 \lhd s^3$ and s^3 not being an end node we already have $1? = \mathcal{P}(s^2, s^3)$.

It seems possible that by changing the critical rule the PDL-calculus can be changed into a DPDL-calculus and by adding another rule into a CPDL-calculus (both logics are also decidable [5]). In particular then the following questions are interesting:

- Does the Interpolation Theorem also hold for DPDL and/or CPDL, and can it be proven in the same way as for PDL using a suitable tableau-calculus?

↓ page 60

Appendix

List of all rules

The rules of the PDL-calculus:

$$(\neg)\ \frac{X;\neg\neg P}{X;P} \qquad (\wedge)\ \frac{X;P\wedge Q}{X;P;Q} \qquad (\neg\wedge)\ \frac{X;\neg(P\wedge Q)}{X;\neg P \ \mid\ X;\neg Q}$$

$$(\neg\cup)\ \frac{X;\neg[a\cup b]P}{X;\neg[a]P \ \mid\ X;\neg[b]P} \qquad (\neg?)\ \frac{X;\neg[Q?]P}{X;Q;\neg P} \qquad (\neg;)\ \frac{X;\neg[a;b]P}{X;\neg[a][b]P}$$

$$(\cup)\ \frac{X;[a\cup b]P}{X;[a]P;[b]P} \qquad (?)\ \frac{X;[Q?]P}{x;\neg Q \ \mid\ X;P} \qquad (;)\ \frac{X;[a;b]P}{X;[a][b]P}$$

$$(\neg n)\ \frac{X;\neg[a^*]P}{X;\neg P \ \mid\ X;\neg[a][a^{(n)}]P} \qquad (n)\ \frac{X;[a^*]P}{X;P;[a][a^{(n)}]P}$$

$$(M+)\ \frac{X;\neg[a_0]\dots[a_n]P}{X;\neg[a_0]\dots[a_n]P^P}\quad X \text{ free} \qquad \text{the } \textit{loading rule},$$

$$(M-)\ \frac{X;\neg[a]P^R}{X;\neg[a]P} \qquad \text{the } \textit{liberation rule},$$

$$(At)\ \frac{X;\neg[A]P^R}{X_A;\neg P^{R\setminus P}} \qquad \text{the } \textit{critical} \text{ rule}.$$

$$(\neg\cup)\ \frac{X;\neg[a\cup b]P^R}{X;\neg[a]P^R \ \mid\ X;\neg[b]P^R} \qquad (\neg;)\ \frac{X;\neg[a;b]P^R}{X;\neg[a][b]P^R}$$

$$(\neg n)\ \frac{X;\neg[a^*]P^R}{X;\neg P^{R\setminus P} \ \mid\ X;\neg[a][a^{(n)}]P^R} \qquad (\neg?)\ \frac{X;\neg[Q?]P^R}{X;Q;\neg P^{R\setminus P}}$$

The rules for the construction of extended tableaus:

$$(*;)\ \frac{X;[a^*;b]P}{X;[b]P \ \mid\ X;[a][a^*;b]P} \qquad (?;)\ \frac{X;[1?;a]P}{X;[a]P}$$

$$(F-)\ \frac{X;Y}{X}\quad Y \text{ free} \qquad (1?)\ \frac{X;[1?]P}{X;P}$$

↓ page 61

References

[1] M.BEN-ARI, J.Y.HALPERN, A.PNUELI, *Deterministic Propositional Dynamic Logic: Finite Models, Complexity, and Completeness*, Journal of Computer and System Sciences 25 (1982), 402–417 https://doi.org/10.1016/0022-0000(82)90018-6

[2] F.BERMAN, M.PATERSON, *Propositional Dynamic Logic is weaker without Tests*, Theoretical Computer Science 16 (1981), 321–328 https://doi.org/10.1016/0304-3975(81)90102-X

[3] K.A.BOWEN, *Interpolation in Loop-free Logic*, Studia Logica 39 (1980), 297–310 https://www.jstor.org/stable/20014987

[4] M.J.FISCHER, R.E.LADNER, *Propositional Dynamic Logic of Regular Programs*, Journal of Computer and System Sciences 19 (1979), 194–211 https://doi.org/10.1016/0022-0000(79)90046-1

[5] D.HAREL, *Dynamic Logic*, in "Handbook of Philosophical Logic" Vol.II, 497–604, (D.GABBAY, F.GUENTHER eds.), Reidel Dordrecht 1984 https://doi.org/10.1007/978-94-009-6259-0_10

[6] G.H.MÜLLER, W.RAUTENBERG (eds.) Ω-*Bibliography of Mathematical Logic*, Vol II, Springer Berlin 1987 Subtitle: *Non-Classical Logics*, https://www.springer.com/de/book/9783662090572

[7] V.R.PRATT, *Semantical Considerations on Floyd-Hoare Logic*, 17th IEEE Symposium on Foundations of Computer Science (1976), 109–121 https://doi.org/10.1109/SFCS.1976.27

[8] V.R.PRATT, *A Practical Decision Method for Propositional Dynamic Logic*, ACM Symposium on Theory of Computing 10 (1978), 326–337 https://doi.org/10.1145/800133.804362

[9] V.R.PRATT, *Models of Program Logics*, 20th IEEE Symposium on Foundations of Computer Science (1979), 115–122 https://doi.org/10.1109/SFCS.1979.24

[10] V.R.PRATT, *A Near-Optimal Method for Reasoning about Action*, Journal of Computer and System Sciences 20 (1980), 231–254 https://doi.org/10.1016/0022-0000(80)90061-6

[11] W.RAUTENBERG, *Klassische und Nichtklassische Aussagenlogik*, Wiesbaden 1979 https://doi.org/10.1007/978-3-322-85796-5

↓ page 62

[12] W.RAUTENBERG *Modal Tableau Calculi and Interpolation*, J. Philosophical Logic 12 (1983), 402–423 https://www.jstor.org/stable/30226284

[13] M.Y.VARDI, P.WOLPER, *Automata-Theoretic Techniques for Modal Logic of Programs*, Journal of Computer and System Sciences 32 (1986), 183–221 https://doi.org/10.1016/0022-0000(86)90026-7

Additional references:

[14] P. BLACKBURN, M. de RIJKE, Y. VENEMA, *Modal Logic*, Cambridge University Press (2001), ISBN 978-0-521-52714-9

List of Symbols

$p, q, \ldots$	propositional variables
$\mathcal{F}_0$	the set of propositional variables
$P, Q, \ldots$	formulas
$\mathcal{F}$	the set of all formulas
$X, Y, Z, \ldots$	sets of formulas
$A, B, \ldots$	atomic programs
$\mathcal{P}_0$	the set of atomic programs
$a, b, \ldots$	programs
$\xrightarrow{a}$	program relation between states or sets of formulas
$\mathcal{P}$	the set of all programs
$\mathcal{T}$	a tableau
$s, t, \ldots$	nodes in tableaus
$x, x_1, x_2, \ldots$	functions mapping nodes to (pairs of) sets of formulas
$\lhd$	successor relation (Definition 6)
$\leq$	reflexive transitive closure of $\lhd$
$<$	$s \leq t$ and $s \neq t$
$I, I_1, I_2, \ldots$	interpolants
f	the function mapping $a^{(n)}$ to a^* etc.

Selected Vocabulary

Deutsch	English	meaning/example	defined at
einfach	simple		Definition 9
markiert	marked	P^P	
belastet	loaded	contains a P^P	on page 19
frei	free	not loaded	on page 19
n-Formel	n-formula	contains an $a^{(n)}$	Definition 10
normal	normal	not an n-formula	Definition 10
Fortsetzung	Continuation		
der Gestalt	of the form		
Rechenwerk	CPU		