A Non-wellfounded, Labelled Proof System for Propositional Dynamic Logic

Simon Docherty, University College London
Reuben N. S. Rowe, Royal Holloway University of London

TABLEAUX 2019
2nd–5th September 2019, Middlesex University, London, UK

What is Dynamic Logic?

Dynamic Logic was introduced by Pratt (1976)

- Reasoning about program executions (i.e. their dynamics)
- A modal logic (programs are modal operators)

$$x \geq 3 \rightarrow [x := x + 1](x \geq 4)$$

What is Dynamic Logic?

Dynamic Logic was introduced by Pratt (1976)

- Reasoning about program executions (i.e. their dynamics)
- A modal logic (programs are modal operators)

$$x \geq 3 \rightarrow [x := x + 1](x \geq 4)$$

Intuitively, for a program p and assertion φ:

$[p]\varphi$ means φ holds after *all* (terminating) executions of p

$\langle p \rangle \varphi$ means there is *some* execution of p after which φ holds

The Language of Programs

Programs are constructed from:

- A set of basic programs (e.g. $x := x + 1$)
- Sequential composition $p \,;\, q$
- Non-deterministic choice $p \cup q$
- Iteration p^*

The Language of Programs

Programs are constructed from:

- A set of basic programs (e.g. $x := x + 1$)
- Sequential composition $p\,;q$
- Non-deterministic choice $p \cup q$
- Iteration p^*
- For any formula φ, the test $\varphi?$ is a program

The Language of Programs

Programs are constructed from:

- A set of basic programs (e.g. $x := x + 1$)
- Sequential composition $p \,;\, q$
- Non-deterministic choice $p \cup q$
- Iteration p^*
- For any formula φ, the test $\varphi?$ is a program

So, programs form a Kleene Algebra (with tests)

The Language of Programs

Programs are constructed from:

- A set of basic programs (e.g. $x := x + 1$)
- Sequential composition $p \,;\, q$
- Non-deterministic choice $p \cup q$
- Iteration p^*
- For any formula φ, the test $\varphi?$ is a program

So, programs form a Kleene Algebra (with tests)

- Various extensions: converse p^-, intersection $p \cap q$, etc.

Relational (Kripke) Semantics of Dynamic Logic

Basic programs are accessibility relations on (memory) states $s \in \mathcal{S}$

$$[\![x := x + 1]\!] = \{(x \mapsto 0, x \mapsto 1), (x \mapsto 1, x \mapsto 2), \ldots\}$$

Relational (Kripke) Semantics of Dynamic Logic

Basic programs are accessibility relations on (memory) states $s \in \mathcal{S}$

$$[\![x := x + 1]\!] = \{(x \mapsto 0, x \mapsto 1), (x \mapsto 1, x \mapsto 2), \ldots\}$$

Formulas are interpreted as sets of states

$$[\![\, \langle p \rangle \varphi \,]\!] = \{s \mid (s, s') \in [\![p]\!] \wedge s' \in [\![\varphi]\!]\}$$

$$[\![\, [p] \varphi \,]\!] = \neg [\![\, \langle p \rangle \neg \varphi \,]\!] = \mathcal{S} \setminus \{s \mid (s, s') \in [\![p]\!] \wedge s' \in \mathcal{S} \setminus [\![\varphi]\!]\}$$

Relational (Kripke) Semantics of Dynamic Logic

Basic programs are accessibility relations on (memory) states $s \in \mathcal{S}$

$$[\![x := x + 1]\!] = \{(x \mapsto 0, x \mapsto 1), (x \mapsto 1, x \mapsto 2), \ldots\}$$

Formulas are interpreted as sets of states

$$[\![\, \langle p \rangle \varphi \,]\!] = \{s \mid (s, s') \in [\![p]\!] \wedge s' \in [\![\varphi]\!]\}$$
$$[\![\, [p]\varphi \,]\!] = \neg [\![\, \langle p \rangle \neg \varphi \,]\!] = \mathcal{S} \setminus \{s \mid (s, s') \in [\![p]\!] \wedge s' \in \mathcal{S} \setminus [\![\varphi]\!]\}$$

Relational interpetation of the program algebra is standard

$$[\![p \,;\, q]\!] = [\![p]\!] \circ [\![q]\!] \qquad [\![p \cup q]\!] = [\![p]\!] \cup [\![q]\!] \qquad [\![p^*]\!] = \bigcup_{n \geq 0} [\![p]\!]^n$$

Relational (Kripke) Semantics of Dynamic Logic

Basic programs are accessibility relations on (memory) states $s \in \mathcal{S}$

$$[\![x := x+1]\!] = \{(x \mapsto 0, x \mapsto 1), (x \mapsto 1, x \mapsto 2), \ldots\}$$

Formulas are interpreted as sets of states

$$[\![\, \langle p \rangle \varphi \,]\!] = \{s \mid (s, s') \in [\![p]\!] \wedge s' \in [\![\varphi]\!]\}$$
$$[\![\, [p]\varphi \,]\!] = \neg [\![\, \langle p \rangle \neg \varphi \,]\!] = \mathcal{S} \setminus \{s \mid (s, s') \in [\![p]\!] \wedge s' \in \mathcal{S} \setminus [\![\varphi]\!]\}$$

Relational interpetation of the program algebra is standard

$$[\![p \,;\, q]\!] = [\![p]\!] \circ [\![q]\!] \qquad [\![p \cup q]\!] = [\![p]\!] \cup [\![q]\!] \qquad [\![p^*]\!] = \bigcup_{n \geq 0} [\![p]\!]^n$$

But tests introduce a mutual recursion: $[\![\varphi?]\!] = \{(s, s) \mid s \in [\![\varphi]\!]\}$

The Influence of Dynamic Logic

Lots of variants and extensions:

- Games (Parikh, ’83)
- Natural language (Groenendijk & Stokhof, ’91)
- Knowledge representation (De Giacomo & Lenzarini, ’94)
- XML (Afanasiev Et Al, 2005)
- Cyber-physical systems (Platzer, 2008)
- Epistemic reasoning for agents (Patrick Girard Et Al, 2012)
- etc.

What is Propositional Dynamic Logic?

Fischer & Ladner (1979) first studied the propositional fragment

- Only abstract propositional programs
- No quantification

What is Propositional Dynamic Logic?

Fischer & Ladner (1979) first studied the propositional fragment

- Only abstract propositional programs
- No quantification

PDL is the logic of (regular) programs

$$[\alpha^*]((\varphi \to [\alpha]\neg\varphi) \land (\neg\varphi \to [\alpha]\varphi)) \leftrightarrow [(\alpha\,;\alpha)^*]\varphi \lor [(\alpha\,;\alpha)^*]\neg\varphi$$

What is Propositional Dynamic Logic?

Fischer & Ladner (1979) first studied the propositional fragment

- Only abstract propositional programs
- No quantification

PDL is the logic of (regular) programs

$$[\alpha^*]((\varphi \to [\alpha]\neg\varphi) \land (\neg\varphi \to [\alpha]\varphi)) \leftrightarrow [(\alpha\,;\alpha)^*]\varphi \lor [(\alpha\,;\alpha)^*]\neg\varphi$$

$$\texttt{if}\ \varphi\ \texttt{then}\ \alpha\ \texttt{else}\ \beta \stackrel{\text{def}}{=} (\varphi?\,;\alpha) \cup (\neg\varphi?\,;\beta)$$

$$\texttt{while}\ \varphi\ \texttt{do}\ \alpha \stackrel{\text{def}}{=} (\varphi?\,;\alpha)^*\,;\neg\varphi?$$

PDL: Main Properties and Results

- Small model property
- Satisfiability **EXPTIME**-complete
- Finitely axiomatisable

(K)	$\vdash [\alpha](\varphi \to \psi) \to ([\alpha]\varphi \to [\alpha]\psi)$	(Test)	$\vdash [\psi?]\varphi \leftrightarrow (\psi \to \varphi)$
(Distributivity)	$\vdash [\alpha](\varphi \land \psi) \leftrightarrow ([\alpha]\varphi \land [\alpha]\psi)$	(Fixed Point)	$\vdash \varphi \land [\alpha][\alpha^*]\varphi \leftrightarrow [\alpha^*]\varphi$
(Choice)	$\vdash [\alpha \cup \beta]\varphi \leftrightarrow [\alpha]\varphi \land [\beta]\varphi$	(Induction)	$\vdash \varphi \land [\alpha^*](\varphi \to [\alpha]\varphi) \to [\alpha^*]\varphi$
(Composition)	$\vdash [\alpha ; \beta]\varphi \leftrightarrow [\alpha][\beta]\varphi$	(Necessitation)	from $\vdash \varphi$ infer $\vdash [\alpha]\varphi$

Dual axioms for $\langle\alpha\rangle$ (if taken as a primitive)

PDL: Main Properties and Results

- Small model property
- Satisfiability **EXPTIME**-complete
- Finitely axiomatisable

(K)	$\vdash [\alpha](\varphi \to \psi) \to ([\alpha]\varphi \to [\alpha]\psi)$	(Test)	$\vdash [\psi?]\varphi \leftrightarrow (\psi \to \varphi)$
(Distributivity)	$\vdash [\alpha](\varphi \land \psi) \leftrightarrow ([\alpha]\varphi \land [\alpha]\psi)$	(Fixed Point)	$\vdash \varphi \land [\alpha][\alpha^*]\varphi \leftrightarrow [\alpha^*]\varphi$
(Choice)	$\vdash [\alpha \cup \beta]\varphi \leftrightarrow [\alpha]\varphi \land [\beta]\varphi$	(Induction)	$\vdash \varphi \land [\alpha^*](\varphi \to [\alpha]\varphi) \to [\alpha^*]\varphi$
(Composition)	$\vdash [\alpha\,;\beta]\varphi \leftrightarrow [\alpha][\beta]\varphi$	(Necessitation)	from $\vdash \varphi$ infer $\vdash [\alpha]\varphi$

Dual axioms for $\langle\alpha\rangle$ (if taken as a primitive)

- But not compact $\quad \{\neg\varphi, [\alpha]\neg\varphi, [\alpha\,;\alpha]\neg\varphi, [\alpha\,;\alpha\,;\alpha]\neg\varphi, \ldots\} \not\models \langle\alpha^*\rangle\varphi$

Proof Systems for PDL

Tableaux-based systems:

- De Giacomo & Massacci, 2000
- Goré & Widmann, 2009

Sequent-based with ω-rules/infinite contexts:

- Renardel de Lavalette Et Al, 2008
- Hill & Poggiolesi, 2010
- Fritella Et Al, 2014

Our Goal: A Satisfactory Proof Theory

A robust, structural proof theory for PDL and PDL-type logics

- Analytic and finitary (i.e. automatable!)
- Uniform, modular and extensible

Our Goal: A Satisfactory Proof Theory

A robust, structural proof theory for PDL and PDL-type logics

- Analytic and finitary (i.e. automatable!)
- Uniform, modular and extensible

We combine two methodologies

- Labelled sequent calculus
- Non-wellfounded proof theory

Why Labelled Sequent Calculus?

Modularly capture a range of modal logics (Negri, 2005) using:

- Labelled formulas $x : \varphi$ and relational statements $x\ R\ y$
- Proof rules expressing the meaning of modalities

$$\frac{y : \varphi, x : \Box\varphi, x\ R\ y, \Gamma \Rightarrow \Delta}{x : \Box\varphi, x\ R\ y, \Gamma \Rightarrow \Delta} \qquad \frac{x\ R\ y, \Gamma \Rightarrow \Delta, y : \varphi}{\Gamma \Rightarrow \Delta, x : \Box\varphi}\ (y \text{ fresh})$$

- Proof rules characterising different (geometric) frame properties, e.g.

$$\text{(symm):}\ \frac{y\ R\ x, x\ R\ y, \Gamma \Rightarrow \Delta}{x\ R\ y, \Gamma \Rightarrow \Delta} \qquad \text{(trans):}\ \frac{x\ R\ z, x\ R\ y, y\ R\ z, \Gamma \Rightarrow \Delta}{x\ R\ y, y\ R\ z, \Gamma \Rightarrow \Delta}$$

- Even possible to capture some non-modally definable frame properties

Why Non-wellfounded Proofs?

They allow us to tame (inductive) infinitary behaviour

- Allow derivations to be infinitely tall (vs. wide) — not generally sound!
- Distinguish 'good' derivations with a global trace condition
- Restrict to (finitely representable) cyclic proofs

Why Non-wellfounded Proofs?

They allow us to tame (inductive) infinitary behaviour

- Allow derivations to be infinitely tall (vs. wide) — not generally sound!
- Distinguish ‘good’ derivations with a global trace condition
- Restrict to (finitely representable) cyclic proofs

Examples of non-wellfounded proof theories include:

- FOL + Inductive Definitions (Brotherston & Simpson)
- FOL over Herbrand models (Cohen, R, Zohar)
- Linear Logic with fixed points
  (Fortier & Santocanale, Baelde/Saurin/Doumane/Nollet/Tasson)
- Kleene/Action Algebra (Das & Pous)

Our Non-wellfounded, Labelled Sequent Calculus for PDL

- Relational statements $x\ R_a\ y$ refer to atomic programs a
- Rules for atomic modalities à la Negri

$$(\Box L): \frac{y : \varphi, \Gamma \Rightarrow \Delta}{x : [a]\varphi, x\ R_a\ y, \Gamma \Rightarrow \Delta} \qquad (\Box R): \frac{x\ R_a\ y, \Gamma \Rightarrow \Delta, y : \varphi}{\Gamma \Rightarrow \Delta, x : [a]\varphi}\ (y \text{ fresh})$$

Our Non-wellfounded, Labelled Sequent Calculus for PDL

- Relational statements $x\ R_a\ y$ refer to atomic programs a
- Rules for atomic modalities à la Negri

$$(\Box L):\ \frac{y : \varphi, \Gamma \Rightarrow \Delta}{x : [a]\varphi, x\ R_a\ y, \Gamma \Rightarrow \Delta} \qquad (\Box R):\ \frac{x\ R_a\ y, \Gamma \Rightarrow \Delta, y : \varphi}{\Gamma \Rightarrow \Delta, x : [a]\varphi}\ (y \text{ fresh})$$

- Decompose non-atomic modalities as per semantics, e.g.

$$(\cup L):\ \frac{x : [\alpha]\varphi, x : [\beta]\varphi, \Gamma \Rightarrow \Delta}{x : [\alpha \cup \beta]\varphi, \Gamma \Rightarrow \Delta} \qquad (\cup R):\ \frac{\Gamma \Rightarrow \Delta, x : [\alpha]\varphi \quad \Gamma \Rightarrow \Delta, x : [\beta]\varphi}{\Gamma \Rightarrow \Delta, x : [\alpha \cup \beta]\varphi}$$

Our Non-wellfounded, Labelled Sequent Calculus for PDL

- Relational statements $x\ R_a\ y$ refer to atomic programs a
- Rules for atomic modalities à la Negri

$$(\Box L): \frac{y : \varphi, \Gamma \Rightarrow \Delta}{x : [a]\varphi, x\ R_a\ y, \Gamma \Rightarrow \Delta} \qquad (\Box R): \frac{x\ R_a\ y, \Gamma \Rightarrow \Delta, y : \varphi}{\Gamma \Rightarrow \Delta, x : [a]\varphi}\ (y \text{ fresh})$$

- Decompose non-atomic modalities as per semantics, e.g.

$$(\cup L): \frac{x : [\alpha]\varphi, x : [\beta]\varphi, \Gamma \Rightarrow \Delta}{x : [\alpha \cup \beta]\varphi, \Gamma \Rightarrow \Delta} \qquad (\cup R): \frac{\Gamma \Rightarrow \Delta, x : [\alpha]\varphi \quad \Gamma \Rightarrow \Delta, x : [\beta]\varphi}{\Gamma \Rightarrow \Delta, x : [\alpha \cup \beta]\varphi}$$

- Rules for iteration express its nature as a fixed point

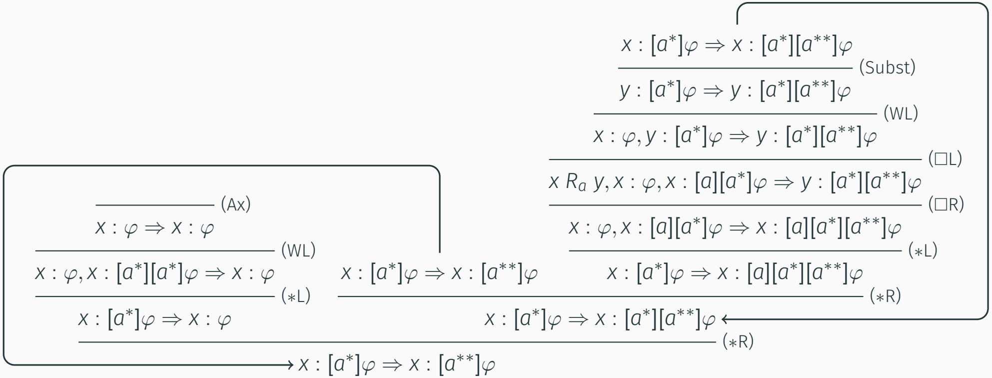
$$(*L): \frac{x : \varphi, x : [\alpha][\alpha^*]\varphi, \Gamma \Rightarrow \Delta}{x : [\alpha^*]\varphi, \Gamma \Rightarrow \Delta} \qquad (*R): \frac{\Gamma \Rightarrow \Delta, x : \varphi \quad \Gamma \Rightarrow \Delta, x : [\alpha][\alpha^*]\varphi}{\Gamma \Rightarrow \Delta, x : [\alpha^*]\varphi}$$

A ‘Bad’ Non-wellfounded Derivation

$$
\dfrac{
\dfrac{
\dfrac{\begin{array}{c}\vdots \\ \Rightarrow x : [\alpha^*]\varphi, x : [\alpha^*]\varphi\end{array}}{\Rightarrow x : [\alpha^*]\varphi}\,(\mathrm{CR})
}{\Rightarrow x : [\alpha^*]\varphi, x : \varphi}\,(\mathrm{WR})
\qquad
\dfrac{
\dfrac{\begin{array}{c}\vdots \\ \Rightarrow x : [\alpha^*]\varphi, x : [\alpha^*]\varphi\end{array}}{\Rightarrow x : [\alpha^*]\varphi}\,(\mathrm{CR})
}{\Rightarrow x : [\alpha^*]\varphi, x : [\alpha][\alpha^*]\varphi}\,(\mathrm{WR})
}{
\dfrac{\Rightarrow x : [\alpha^*]\varphi, x : [\alpha^*]\varphi}{\Rightarrow x : [\alpha^*]\varphi}\,(\mathrm{CR})
}\,(\ast\mathrm{R})
$$

We trace (possibly nested) modalities on the right-hand side

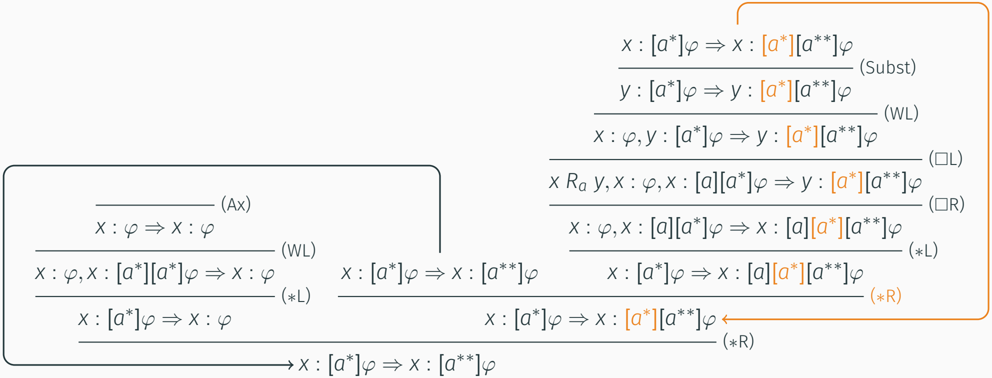
- They must be unfolded infinitely often along infinite paths



'Good' Proofs: The Global Trace Condition

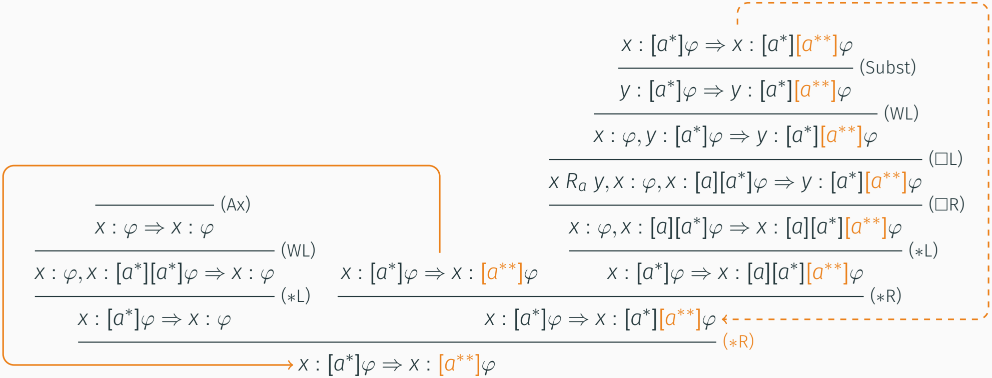
We trace (possibly nested) modalities on the right-hand side

- They must be unfolded infinitely often along infinite paths



We trace (possibly nested) modalities on the right-hand side

- They must be unfolded infinitely often along infinite paths

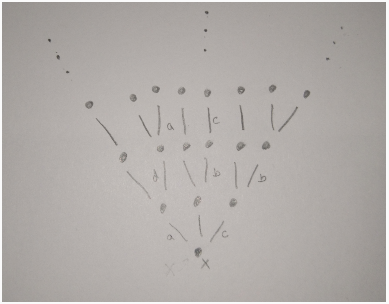


Soundness

Theorem

$\Gamma \Rightarrow \Delta$ is valid if there is a non-wellfounded proof deriving it

- Traced modalities $\Gamma \Rightarrow \Delta, x : [\alpha_1] \dots [\alpha_n][\beta^*]\varphi$ identify particular substructures in countermodels:

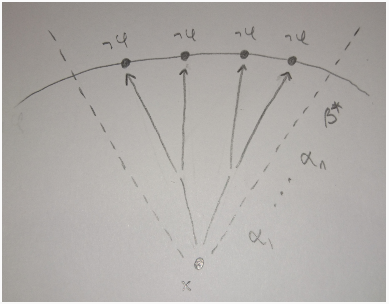


Soundness

Theorem

$\Gamma \Rightarrow \Delta$ is valid if there is a non-wellfounded proof deriving it

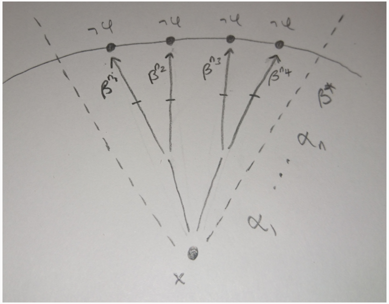
- Traced modalities $\Gamma \Rightarrow \Delta, x : [\alpha_1] \ldots [\alpha_n][\beta^*]\varphi$ identify particular substructures in countermodels:



Theorem

$\Gamma \Rightarrow \Delta$ is valid if there is a non-wellfounded proof deriving it

- Traced modalities $\Gamma \Rightarrow \Delta, x : [\alpha_1] \ldots [\alpha_n][\beta^*]\varphi$ identify particular substructures in countermodels:

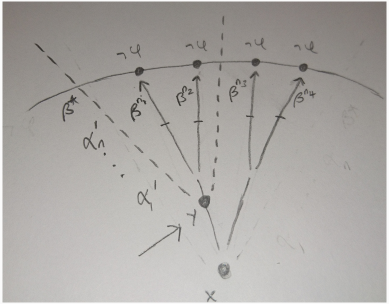


Soundness

Theorem

$\Gamma \Rightarrow \Delta$ is valid if there is a non-wellfounded proof deriving it

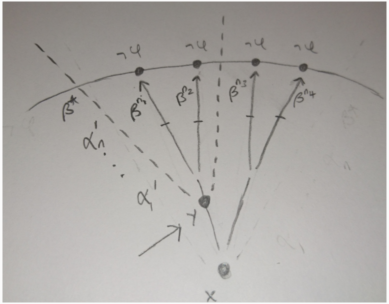
- Traced modalities $\Gamma \Rightarrow \Delta, x : [\alpha_1] \dots [\alpha_n][\beta^*]\varphi$ identify particular substructures in countermodels:



Theorem

$\Gamma \Rightarrow \Delta$ is valid if there is a non-wellfounded proof deriving it

- Traced modalities $\Gamma \Rightarrow \Delta, x : [\alpha_1] \ldots [\alpha_n][\beta^*]\varphi$ identify particular substructures in countermodels:



- Cyclic proofs capture an infinite-descent style proof by contradiction.

Completeness

Theorem

There is a cut-free non-wellfounded proof of each valid $\Gamma \Rightarrow \Delta$

Theorem

There is a cut-free non-wellfounded proof of each valid $\Gamma \Rightarrow \Delta$

Lemma

The axioms characterising PDL have cyclic proofs

Lemma (Necessitation)

There is a cyclic derivation simulating the rule

$$\frac{x : \varphi_1, \ldots, x : \varphi_n \Rightarrow x : \psi}{x : [\alpha]\varphi_1, \ldots, x : [\alpha]\varphi_n \Rightarrow x : [\alpha]\psi}$$

Completeness

Theorem

There is a cut-free non-wellfounded proof of each valid $\Gamma \Rightarrow \Delta$

Lemma

The axioms characterising PDL have cyclic proofs

Lemma (Necessitation)

There is a cyclic derivation simulating the rule

$$\frac{x : \varphi_1, \dots, x : \varphi_n \Rightarrow x : \psi}{x : [\alpha]\varphi_1, \dots, x : [\alpha]\varphi_n \Rightarrow x : [\alpha]\psi}$$

Theorem

If φ *is a PDL theorem, there is a cyclic proof deriving* $\Rightarrow x : \varphi$

Proof Search for Test-free sequents

We propose the following proof-search strategy:

- Apply (invertible) logical rules as much as possible
  - But do not allow traces to progress more than once
  - For test-free sequents, this terminates
- Close open leaves with axioms where possible
- Apply a series of validity-preserving weakenings
- Repeat process for any remaining open leaves

All formulas that appear are in the Fischer-Ladner closure of the end sequent

Conjecture

The number of distinct labels appearing in a sequent is bounded

Future Work

- Prove cut-free regular completeness results (also for tests?)
- Demonstrate capture of different frame conditions
- Incorporate additional constructs in the program algebra
    - Converse, Intersection
- Extend to capture other modal fixpoints (temporal, common knowledge)
- Derive interpolation results from the proof theory
    - cf. Cyclic system and Lyndon interpolation for for GL (Shamkanov, 2014)