Automated Synthesis of Tableau Calculi

Version of August 12, 2011

Renate A. Schmidt and Dmitry Tishkovsky
With contributions from Mohammad Khodadadi

School of Computer Science
The University of Manchester
{renate.schmidt, dmitry.tishkovsky}@manchester.ac.uk

The 23rd European Summer School
in Logic, Language and Information
8–12 August 2011

http://www.cs.man.ac.uk/~dmitry/courses/ESSLLI2011

Tableau-based deduction

- Has a long tradition and is a well established method in automated reasoning
- Approach can be successfully used for a large number of logics:

Classical	propositional logic, first-order logic, second-order logic
Non-classical	modal, description, hybrid, intuitionistic logics, . . . , temporal, propositional dynamic, fixpoint logics, . . .

- Multitude of different tableau approaches:

First-order logic	Smullyan ground sentence tableau	free-variable tableau, connection tableau, disconnection tableau, hypertableau, . . .
Non-classical logics	ground semantic tableau, tableau avoiding reference to semantics, and-or tableau	free-variable tableau

- Many implemented systems

Our interest: Ground semantic tableau calculi for non-classical logics

- Ground semantic tableau calculi = calculi in the style of Smullyan: Operate on labelled formulae, where the labels refer to elements in the semantics
  - Can be used to build models
  - Are easy to understand, learn and teach
  - Are easier to develop and provide more flexibility
  - Basis for decision procedures for many non-classical logics, especially logics with some kind of tree model property
- We became interested in description logics with complex role operators that do not have any kind of tree model property
  - Description logic $\mathcal{ALBO}$ = $\mathcal{ALC}$ with Boolean operators on roles and nominals
  - Can be decided with tableau + unrestricted blocking technique [ISWC07]
- Unrestricted blocking is powerful and generic
- This has given us hope that tableau-based decision procedures can be developed systematically for many logics

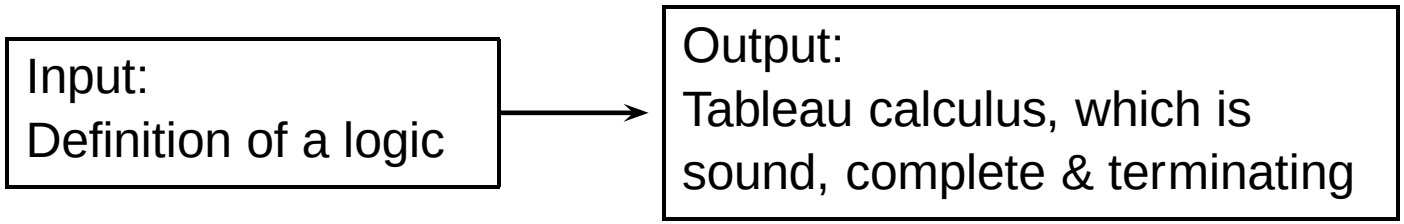
Automation of calculus development

- Existing work for non-classical logics suggests that it should be possible to develop tableau calculi systematically for large classes of logics
  - many variations
  - many similarities
  - important underlying principles tend to be the same
- Questions:
  - Can tableau calculi be developed automatically from the definition of logics?
  - Can soundness and completeness be guaranteed?
  - Can termination be guaranteed?
- No, in each case.

But:

Perhaps it is possible to develop tableau calculi automatically under certain restrictions?

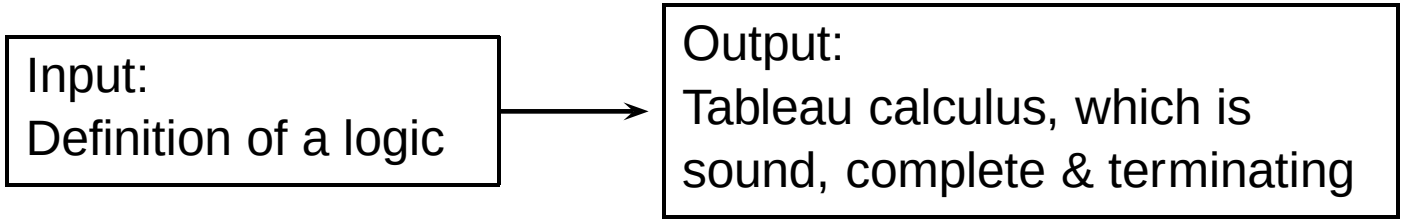
Problem of interest



- Automatic generation of tableau calculi from the definition of a logic + certification of soundness, completeness and termination (if relevant)

- Important issue:
  What respectively are sufficient conditions for soundness, completeness and termination?

Tableau synthesis framework [TABLEAUX09,LMCS11]



Our assumptions

- The given logic is defined by the specification of its semantics
- The specification language of the framework is first-order
- Designed for propositional non-classical logics
- The generated calculi are ground semantic tableau calculi

Aim of the course

- Overall, to introduce and study the tableau synthesis framework
- To describe how users can define their logic in the specification language of the framework (Day 2)
- To introduce the method of generating tableau calculi (Day 3)
- To discuss the conditions and underlying theory of guaranteeing soundness and completeness of the generated tableau calculi (Day 2 & 3)
- To study blocking techniques for ensuring termination, including unrestricted blocking (Day 4)
- To consider various case studies of generating tableau calculi for different logics (throughout & Day 5)

Content

- Treatment is formal and rigorous with emphasis on theoretical foundations and describing and using the framework
- Many examples
- Some basic knowledge of propositional logic and first-order logic is assumed
- Some basic knoweldge of the following is an advantage but not essential:
  - modal logic, description logic, or other non-classical logics
  - tableau-based reasoning
- Content not as detailed as in ordinary lectures
- Thus, please feel free to ask questions anytime !

Main reference

- Schmidt, R. A. and Tishkovsky, D. (2011), Automated Synthesis of Tableau Calculi. *Logical Methods in Computer Science* **7** (2), 1–32. Short version published in *Proc. TABLEAUX 2009*, LNCS 5607, Springer, 310–324 (2009).

Overview of the rest of this session

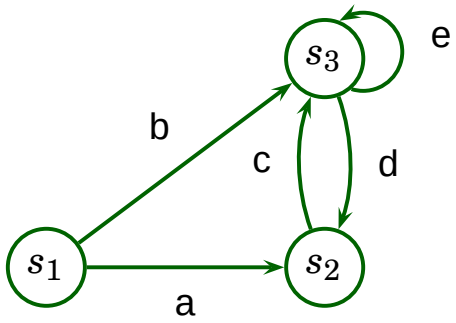
- Introduction of the logics we use as running examples
- Introduction of tableau-based deduction

The logics

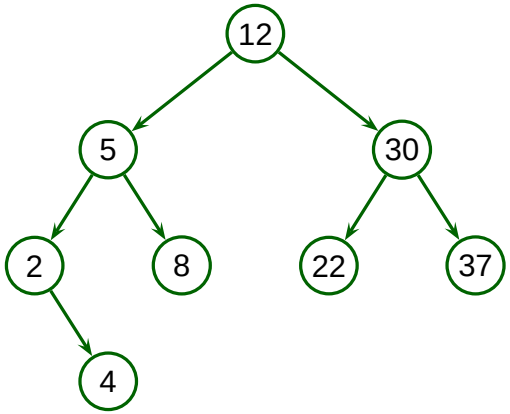
- Basic modal logic K
- Modal logic S4
- Hybrid logic S4$(@)$
- Extension of S4 for determining the admissibility of rules (on Day 5)
- Modal logic of some, all and only, K$_{(m)}(\neg)$ (on Day 5)

Modal logic: Background

- Established field, long history in mathematics and philosophy
- Popular in CS:
  - program specification & verification, NLP,
  - multi-agent systems,
  - description logics, semantic web & ontology reasoning,
  - ...
- There are non-first-order definable MLs, there are undecidable MLs
- But, the commonly used MLs have many good properties:
  - fragments of FOL, decidable, nice computational complexity
  - language is simple & natural, powerful enough to describe useful structures



transition systems



trees

Basic modal logic K

K = propositional logic plus $\Box, \Diamond$

- Formulae: $\phi, \psi \longrightarrow p \mid \bot \mid \neg\phi \mid \phi \vee \psi \mid \Box\phi$

non-empty set of possible worlds

- Semantics: Kripke model $\mathcal{M} = (W, R, v)$

binary relation over W — valuation mapping $v : \mathcal{P} \longrightarrow 2^W$

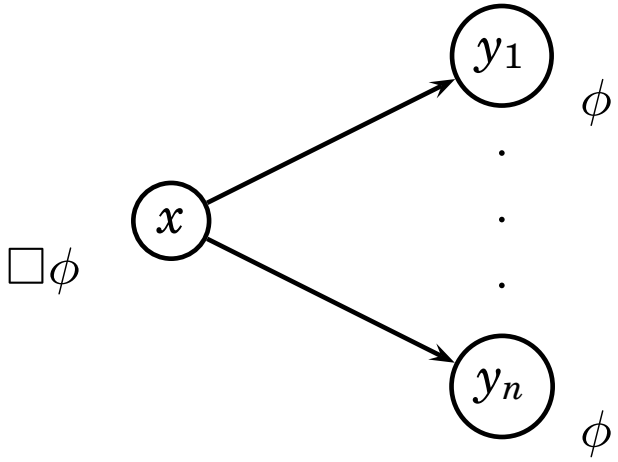
$$
\begin{array}{lll}
\mathcal{M}, x \models p & \text{iff} & x \in v(p) \\
\mathcal{M}, x \not\models \bot & & \\
\mathcal{M}, x \models \neg\phi & \text{iff} & \mathcal{M}, x \not\models \phi \\
\mathcal{M}, x \models \phi \vee \psi & \text{iff} & \mathcal{M}, x \models \phi \text{ or } \mathcal{M}, x \models \psi \\
\mathcal{M}, x \models \Box\phi & \text{iff} & \text{for all } R\text{-successors } y \text{ of } x \ \mathcal{M}, y \models \phi
\end{array}
$$

- Defined operators: $\top \stackrel{\text{def}}{=} \neg\bot$, $\phi \wedge \psi \stackrel{\text{def}}{=} \neg(\neg\phi \vee \neg\psi)$,
  $\phi \rightarrow \psi \stackrel{\text{def}}{=} \neg\phi \vee \psi$ $\Diamond\phi \stackrel{\text{def}}{=} \neg\Box\neg\phi$
- Terminology: *frame* (W, R)

The modal operators

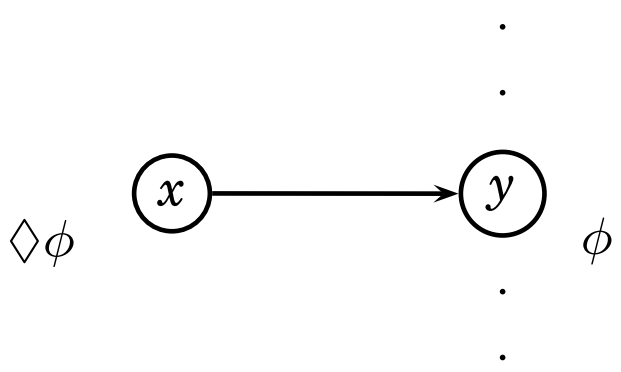
- $\Box$ and $\Diamond$ are the universal and existential quantifiers of modal logic

$\mathcal{M}, x \models \Box\phi$ iff
$\forall y \left(R(x,y) \rightarrow \mathcal{M}, y \models \phi \right)$



ϕ is a *necessity* relative to x

$\mathcal{M}, x \models \Diamond\phi$ iff
$\exists y \left(R(x,y) \wedge \mathcal{M}, y \models \phi \right)$

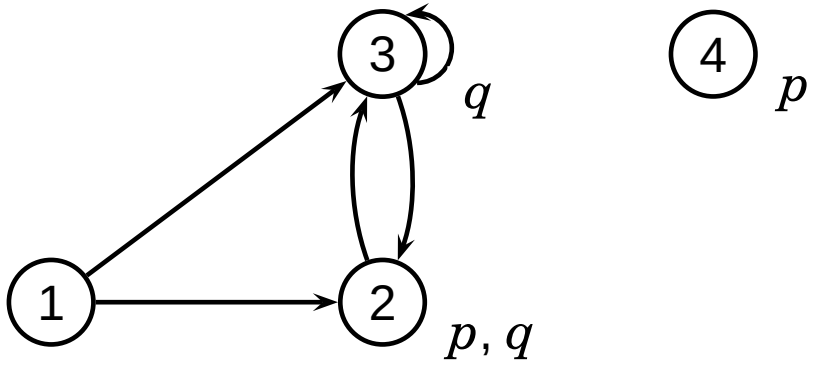


ϕ is a *possibility* relative to x

- Also known respectively as the *necessity operator* and *possibility operator*

Example

Suppose $\mathcal{M} = (W, R, v)$ is the Kripke model depicted by:



We have

$\mathcal{M}, 1 \models \Diamond q$
$\mathcal{M}, 1 \models \Box q$
$\mathcal{M}, 1 \models \Box\Box q?$

$\mathcal{M}, 1 \models \Diamond p$
$\mathcal{M}, 1 \models \neg\Box p$
$\mathcal{M}, 1 \models \Box\Box p?$

Varying the semantics

- It is possible to obtain new modal logics by varying the underlying models

Logic	Frame conditions of R
KT	reflexive: $\forall x \,.\, R(x,x)$
K4	transitive: $\forall x,y,z \,.\, (R(x,y) \wedge R(y,z)) \rightarrow R(x,z)$
S4 = KT4	reflexive transitive
S5 = KT45	reflexive transitive euclidean: $\forall x,y,z \,.\, (R(x,y) \wedge R(x,z)) \rightarrow R(y,z)$
KD45	serial: $\forall x \exists y \,.\, R(x,y)$ transitive euclidean

- All these logics have the same language as K and the semantics is defined exactly as for K but the underlying accessibility relation R satisfies certain *frame conditions*
- In this way infinitely many extensions of K can be defined

Modal logic S4

S4 = K in which the accessibility relation R is a pre-order

- Formulae: $\phi, \psi \longrightarrow p \mid \bot \mid \neg\phi \mid \phi \vee \psi \mid \Box\phi$
- Semantics: Kripke model $\mathcal{M} = (W, R, v)$

$$\begin{array}{lll} \mathcal{M}, x \models p & \text{iff} & x \in v(p) \\ \mathcal{M}, x \not\models \bot & & \\ \mathcal{M}, x \models \neg\phi & \text{iff} & \mathcal{M}, x \not\models \phi \\ \mathcal{M}, x \models \phi \vee \psi & \text{iff} & \mathcal{M}, x \models \phi \text{ or } \mathcal{M}, x \models \psi \\ \mathcal{M}, x \models \Box\phi & \text{iff} & \text{for all } R\text{-successors } y \text{ of } x \ \mathcal{M}, y \models \phi \end{array}$$

- R is reflexive: for all x $(x, x) \in R$
  R is transitive: for all x, y, z
  if $(x, y) \in R$ and $(y, z) \in R$ then $(x, z) \in R$

Extending the language of basic modal logic

- On the other hand, a number of modal logics have languages which extend that of basic modal logic

Multiple $\Box_j$ operators:

$\mathcal{M}, x \models \Box_j \phi$ iff for all R_j-successors y of x $\mathcal{M}, y \models \phi$

Universal modality $[u]$:

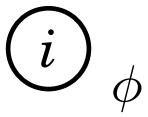
$\mathcal{M}, x \models [u]\phi$ iff for all y $\mathcal{M}, y \models \phi$

Satisfaction operator $@_i$:

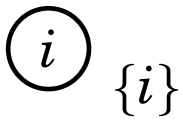
$\mathcal{M}, x \models @_i \phi$ iff $\mathcal{M}, y \models \phi$ where y is the denotation of i

Singleton set operator $\{\cdot\}$:

$\mathcal{M}, x \models \{i\}$ iff x is the denotation of i



stipulating truth at a named world



true at exactly one world

Extending S4 with the at operator

Hybrid logic S4$(@)$ = S4 plus at operator

- Nominals: i
  Formulae: $\phi, \psi \longrightarrow p \mid \bot \mid \neg\phi \mid \phi \vee \psi \mid \Box\phi \mid @_i\phi$

- Semantics: Kripke model $\mathcal{M} = (W, R, v, v_0)$
  valuation mapping $v_0 : \mathcal{N} \longrightarrow W$

$$\begin{array}{lll}
\mathcal{M}, x \models p & \text{iff} & x \in v(p) \\
\mathcal{M}, x \not\models \bot & & \\
\mathcal{M}, x \models \neg\phi & \text{iff} & \mathcal{M}, x \not\models \phi \\
\mathcal{M}, x \models \phi \vee \psi & \text{iff} & \mathcal{M}, x \models \phi \text{ or } \mathcal{M}, x \models \psi \\
\mathcal{M}, x \models \Box\phi & \text{iff} & \text{for all } R\text{-successors } y \text{ of } x \ \mathcal{M}, y \models \phi \\
\mathcal{M}, x \models @_i\phi & \text{iff} & \mathcal{M}, v_0(i) \models \phi
\end{array}$$

- R is reflexive and transitive

Idea of semantic tableaux for propositional logic

Recall, a set N of formulae is *true* iff $\bigwedge N$ is true
iff every formula in N is true

- A set $N \cup \{\phi \wedge \psi\}$ of formulae is true iff $N \cup \{\phi \wedge \psi,\ \phi,\ \psi\}$ is true
- A set $N \cup \{\phi \vee \psi\}$ of formulae is true
  iff $N \cup \{\phi \vee \psi,\ \phi\}$ or $N \cup \{\phi \vee \psi,\ \psi\}$ is true
- Similarly for other connectives
- Continue expansion until two complementary formulae ϕ and $\neg\phi$ are found in a set
  $\Longrightarrow$ inconsistency detected

Example

Is the set $N = \{p \wedge \neg(q \vee \neg r),\ \neg q \vee \neg r\}$ satisfiable ?

$\{p \wedge \neg(q \vee \neg r),\ \neg q \vee \neg r\}$

$\Rightarrow \quad \{p \wedge \neg(q \vee \neg r),\ \neg q \vee \neg r,\ \neg q\}$

or $\quad \{p \wedge \neg(q \vee \neg r),\ \neg q \vee \neg r,\ \neg r\}$

$\Rightarrow \quad \{p \wedge \neg(q \vee \neg r),\ \neg q \vee \neg r,\ \neg q,\ p,\ \neg(q \vee \neg r)\}$

or $\quad \{p \wedge \neg(q \vee \neg r),\ \neg q \vee \neg r,\ \neg r\}$

$\Rightarrow \quad \{p \wedge \neg(q \vee \neg r),\ \neg q \vee \neg r,\ \neg q,\ p,\ \neg(q \vee \neg r),\ \cancel{\neg q},\ \neg\neg r\}$

or $\quad \{p \wedge \neg(q \vee \neg r),\ \neg q \vee \neg r,\ \neg r\}$

$\Rightarrow \quad \{p \wedge \neg(q \vee \neg r),\ \neg q \vee \neg r,\ \neg q,\ p,\ \neg(q \vee \neg r),\ \neg\neg r,\ r\}$

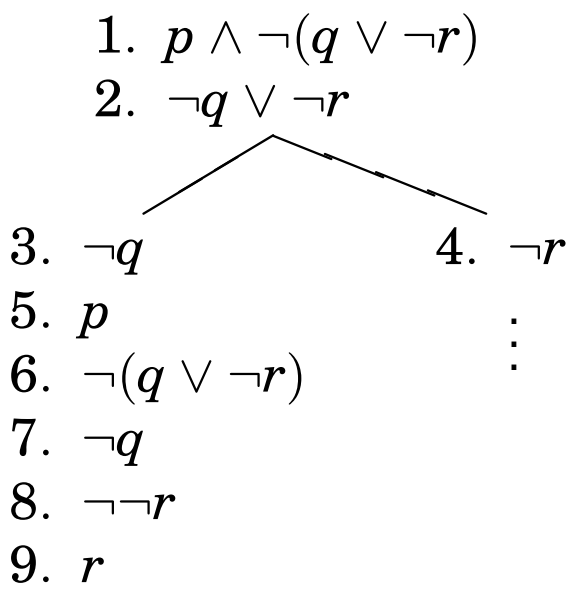
or $\quad \{p \wedge \neg(q \vee \neg r),\ \neg q \vee \neg r,\ \neg r\}$

$\Rightarrow \quad \ldots$

There is a lot of duplication

A tableau derivation for $N = \{p \wedge \neg(q \vee \neg r),\ \neg q \vee \neg r\}$

To avoid duplication, sets are represented as paths of a tree



This tableau is not "fully expanded", however the first "branch" is.
This branch is not "closed", hence the set $\{1, 2\}$ is satisfiable. (These notions will all be defined below.)

General form of tableau rules, and rule application

- The inference rules, called *expansion rules*, have the form

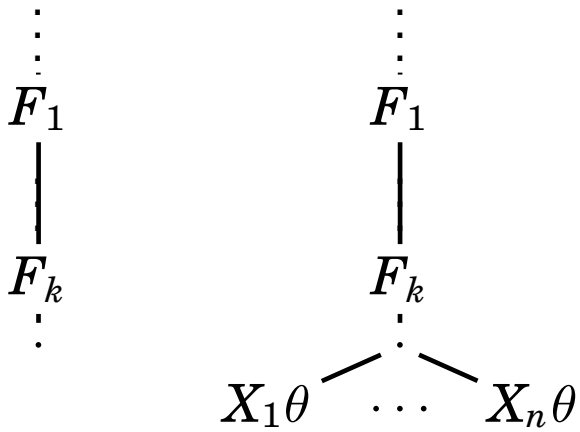
$$\frac{X}{X_1 \mid \ldots \mid X_n}$$

where X, X_i denote either one or more formulae

- Formulae in X are called *premises*, and formulae in X_i are called *conclusions* or *inferred formulae*

$$\frac{E_1, \ldots, E_k}{X_1 \mid \ldots \mid X_n}$$

$E_1\theta = F_1, \ldots, E_k\theta = F_k$ for some θ



- Whenever the premises of the expansion rule matches formulae appearing *anywhere* on a branch, we append the conclusions of the rule at the *leaf* of that branch

Expansion rules for propositional logic

$\neg$ expansion rule:

$$\frac{\neg\neg\phi}{\phi}$$ append ϕ

Expansion of conjunctive formulae:

$$\frac{\phi \wedge \psi}{\phi,\ \ \psi}$$ $$\frac{\neg(\phi \vee \psi)}{\neg\phi,\ \ \neg\psi}$$ append both ϕ and ψ (resp. $\neg\phi$ and $\neg\psi$)

Expansion of disjunctive formulae:

$$\frac{\phi \vee \psi}{\phi \mid \psi}$$ $$\frac{\neg(\phi \wedge \psi)}{\neg\phi \mid \neg\psi}$$ append ϕ and ψ (resp. $\neg\phi$ and $\neg\psi$) in two newly created branches

Closure rule:

$$\frac{\neg\phi,\ \ \phi}{\bot}$$ append $\bot$ and close the branch

Definition of semantic tableau derivation

Let $N = \{F_1, \ldots, F_n\}$ be a set of formulae.

A *semantic tableau (derivation)* for N is a marked (by formulae), finite, unordered tree, inductively defined by:

- The tree consisting of a single branch

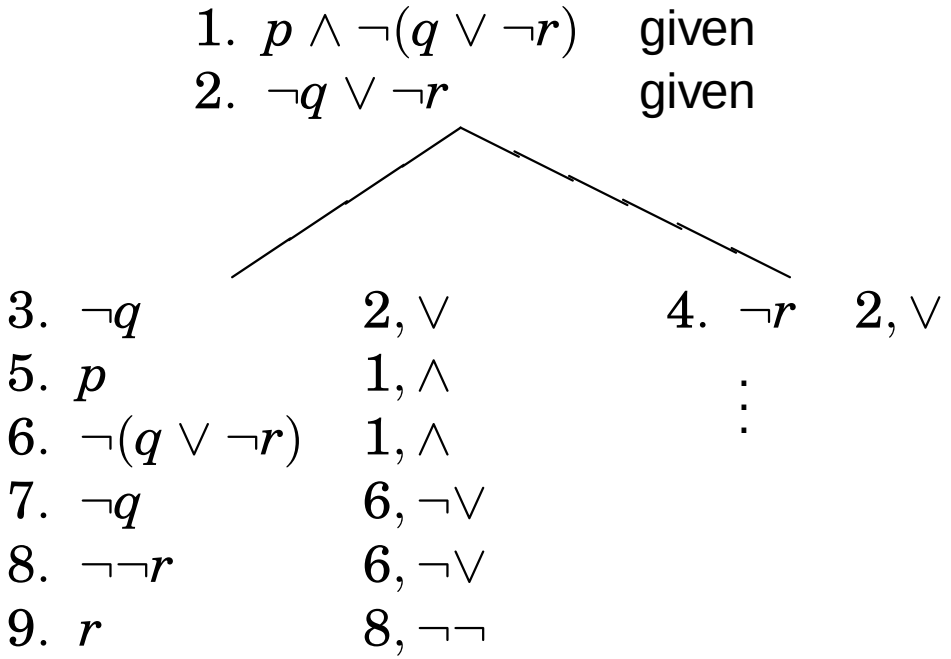
$$F_1$$
$$\vdots$$
$$F_n$$

  is a tableau for N
  (We do not draw edges if nodes have only one successor)

- If T is a tableau for N and T' results from T by applying a rule to T, then T' is a tableau for N

Important assumption: Any rule is applied only *once* to any set of premises on each branch

A formal tableau derivation for $N = \{p \wedge \neg(q \vee \neg r),\ \neg q \vee \neg r\}$



- The left branch is fully expanded and open
- Satisfying assignment found in the left branch:

$$p \mapsto \text{true}, \qquad r \mapsto \text{true}$$

Language of semantic tableau calculi for modal logics

- The tableau rules in the semantic tableau calculi for modal logics operate on labelled formulae
- *Labelled formulae* are defined by:

$$\begin{array}{rcll} \boldsymbol{F} & \longrightarrow & \bot & \text{(contradiction)} \\ & \mid & l : \phi & \text{(\textit{labelled modal formula})} \\ & \mid & (l, l') : R & \text{(\textit{labelled relation})} \end{array}$$

  where l, l' are labels (constants representing worlds), ϕ is a modal formula, R is the accessibility relation

- We assume that the only operators occurring in modal formulae are

$$\bot \qquad \neg \qquad \vee \qquad \Box$$

Tableau calculus for K: Expansion rules

$\neg$ expansion rule:

$$\frac{l : \neg\neg\phi}{l : \phi}$$ append $l : \phi$

$\vee$ expansion rule:

$$\frac{l : \phi \vee \psi}{l : \phi \mid l : \psi}$$ append $l : \phi$ and $l : \psi$ in two newly created branches

$\neg\vee$ expansion rule:

$$\frac{l : \neg(\phi \vee \psi)}{l : \neg\phi,\ \ l : \neg\psi}$$ append both $l : \neg\phi$ and $l : \neg\psi$

Closure rules:

$$\frac{l : \neg\phi,\ \ l : \phi}{\bot}$$ $$\frac{l : \bot}{\bot}$$ append $\bot$ and close the branch

Tableau calculus for K (cont'd): Expansion rules for $\Box$ formulae

$\Box$ expansion rule:

$$\frac{l : \Box\phi, \quad (l,u) : R}{u : \phi}$$

append $u : \phi$

$\neg\Box$ expansion rule:

$$\frac{l : \neg\Box\phi}{(l,u) : R, \quad u : \neg\phi}$$

'create' u and append $(l,u) : R$ and $u : \neg\phi$

where u is a fresh constant

Notes:

- $\neg\Box\phi \equiv \Diamond\neg\phi$
- An inconsequential variation is to use Skolem terms in the $\neg\Box$ rule

A sample tableau derivation

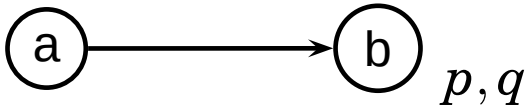
A tableau derivation to determine the K-satisfiability of

$$\{\ \Diamond p,\ \ \Box(p \rightarrow q)\ \}$$

1.	$a : \neg\Box\neg p$	given
2.	$a : \Box(\neg p \vee q)$	given
3.	$(a, b) : R$	$1, \neg\Box$
4.	$b : p$	$1, \neg\Box$
5.	$b : \neg p \vee q$	$2, 3, \Box$

6.	$b : \neg p$	7. $b : q$	$5, \vee$
8.	$\bot$ $4, 6,$ cl		

- The left branch is closed; the right branch is fully expanded and open
- Model found in the right branch:



Important notions

- A *branch* in a tableau is a maximal path from the root to a leaf
- A branch in a tableau is called *closed*, if it contains $\bot$
  Otherwise the branch is called *open*
- A tableau is called *closed*, if all branches are closed
- A branch $\mathcal{B}$ in a tableau is called *fully expanded* (or *complete*), if for each non-atomic formula ϕ, with $l : \phi$ on $\mathcal{B}$ there is a node in $\mathcal{B}$ at which the expansion rule for $l : \phi$ has been applied
- A tableau is called *fully expanded* (or *complete*), if each branch is closed or fully expanded
- Notation: For a tableau calculus *Tab* and a set of labelled formulae N, *Tab*(N) denotes a fully expanded tableau for N

Another tableau derivation

A tableau derivation to determine the K-unsatisfiability of

$$\{\ \Diamond p,\ \ \Box(p \rightarrow q),\ \ \Box\neg q\ \}$$

1.	$a : \neg\Box\neg p$	given
2.	$a : \Box(\neg p \vee q)$	given
3.	$a : \Box\neg q$	given
4.	$(a, b) : R$	$1, \neg\Box$
5.	$b : p$	$1, \neg\Box$
6.	$b : \neg p \vee q$	$2, 4, \Box$

7.	$b : \neg p$		8. $b : q$	$6, \vee$
9.	$\bot$	$5, 7,$ cl	10. $b : \neg q$	$3, 4, \Box$
			11. $\bot$	$8, 10,$ cl

- The left branch is closed; the right branch is closed

Important properties all calculi should have

Any deduction calculus should be proved sound and (refutational) complete

A tableau calculus *Tab* is *sound* iff

$$N \text{ is satisfiable} \implies \text{each fully expanded tableau } \mathit{Tab}(N) \text{ is open}$$

- It suffices to prove: Every rule $X/X_1 \mid \ldots \mid X_n$ is *sound*:

  $$X \text{ is satisfiable} \implies \text{one of the } X_i \text{ is satisfiable}$$

  Then, given a model satisfying N and a tableau *Tab*(N), it is possible to find an open branch in *Tab*(N)

Tab is *refutationally complete* iff

$$N \text{ is unsatisfiable} \implies \text{there is a closed tableau } \mathit{Tab}(N)$$

- It suffices to prove: Given an arbitrary N and an open, fully expanded branch $\mathcal{B}$ in *Tab*(N), construct a model from $\mathcal{B}$ which satisfies N

Soundness and refutational completeness

Theorem 1 (Soundness and refutational completeness of calculus for K)

A set $\{\phi_1, \ldots, \phi_n\}$ of modal formulae is K*-unsatisfiable iff the tableau calculus Tab_{K} can be used to construct a closed tableau for $N = \{a : \phi_1, \ldots, a : \phi_n\}$, where a is a fresh constant*

- More precisely:
  - N is unsatisfiable iff all branches of any tableau $Tab(N)$ constructed for it are closed
  - N is satisfiable iff there is an open, fully expanded branch in some tableau $Tab(N)$ constructed for it
- Proof confluence implies that in fact
  - N is satisfiable iff there is an open, fully expanded branch in *any* tableau $Tab(N)$ constructed for it

Proof confluence

A tableau calculus *Tab* is *proof confluent* iff

N is unsatisfiable $\Longrightarrow$ any partial *Tab*-tableau for N can be expanded into a closed *Tab*(N)

- Essentially there are no dead-ends in the proof search. I.e., which of the potentially many fully expanded tableaux one computes does not matter.

Theorem 2

The calculus $\mathsf{Tab}_{\mathsf{K}}$ is proof confluent for modal formulae

Theorem 3

Smullyan's ground sentence tableau calculus is proof confluent for first-order formulae

Termination

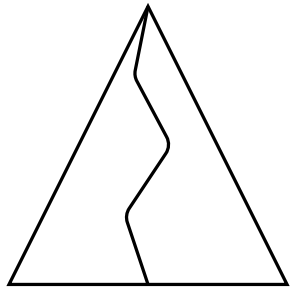
Theorem 4

For K*, every fully expanded tableau* $Tab_{\mathsf{K}}(N)$ *is finitely bounded (for* N *finite)*

- Thus: For modal logic K the construction of fully expanded tableau derivations terminates $\Longrightarrow$ provides decision procedure for K
- The theorem states a strong form of termination

A tableau calculus *Tab* is *strongly terminating* iff for any *finite* set N, every fully expanded tableau *Tab*(N) is finite

Tab is *weakly terminating* iff
for any *finite* set N, every closed tableau *Tab*(N) is finite and every open tableau *Tab*(N) has a finite open branch



Outline

2. Day 2: The specification languages of the framework
   - The object language for specifying the syntax of the given logic
   - The meta-language for specifying the semantics of the logic
   - Semantic specifications
   - Normalised semantic specification
   - Extracting expression orderings from normalised semantic specification
   - Well-defined semantic specification
   - Model structures and first-order definability

In this section we discuss formal languages for representation of syntaxes and semantics of propositional logics. Since semantic specification language is a first-order language we consider a notion of standard first-order models of the language and reduce it to a notion of propositional models usually used for modal, description and other non-classical propositional logics.

Further, we introduce conditions for semantic specifications which will be sufficient for soundness and completeness of generated tableau calculi.

Propositional (Many-Sorted) Logical Languages

Modal logic (K, S4, etc)

- Propositional variables: $p_0, p_1, \ldots$
- Connectives: $\bot$, $\wedge$, $\neg$, $\Box$.

Sorts: Formulae

Hybrid logic, e.g. S4(@)

- Propositional variables: $p_0, p_1, \ldots$
- Nominals: $i_0, i_1, \ldots$
- Connectives: $\bot$, $\wedge$, $\neg$, $\Box$, @.

Sorts: Formulae and nominals

Dynamic modal logic

- Propositional variables: $p_0, p_1, \ldots$
- Relational variables: $r_0, r_1, \ldots$
- Connectives: $\bot$, $\wedge$, $\neg$, $\forall \cdot .$, $\cup$ (relational union), ; (relational composition), etc.

Sorts: Formulae and relations

- Logical languages consist of one or more sorts.
- Each sort is built over its own set of variables with use of connectives.

Primary and Secondary Sorts

- A logical language comprises a single primary sort and possibly several secondary sorts.
- Expressions of the primary sort are “meaningful” expressions which stand for properties of objects in given application domain.

Modal, Hybrid, Dynamic logic: sort of formulae.

- Expressions of secondary sorts are only allowed to form other expressions.

Hybrid logic: sort of nominals.

Dynamic logic: sorts of relations.

Running Example: S4 with @ Operator

Syntax

- S4(@) is an extension of modal logic S4 with nominals and the @-connective.
- S4(@) is a sublogic of the hybrid S4 where nominals are allowed to appear only in the @-connective.
- S4(@) is built over
  - a countable set of nominals $\mathcal{N} = \{i, j, i_0, i_1, \ldots\}$ and
  - a countable set of propositional variables $\mathcal{P} = \{p, q, p_0, p_1, \ldots\}$
- Sorts of S4(@):
  - Nominals i, j
  - Formulae $\phi, \psi \stackrel{\text{def}}{=} \bot \mid p \mid \neg\phi \mid \phi \vee \psi \mid \Box\phi \mid @_i\phi$.
- Connectives $\top$, $\wedge$, $\rightarrow$ and $\Diamond$ are definable connectives, e.g. $\Diamond\phi \stackrel{\text{def}}{=} \neg\Box\neg\phi$.

Running Example: S4 with @ Operator

Sematics

Kripke model $\mathcal{M} = (W, R, \nu, \nu_0)$

- $R \subseteq W^2$, $\nu : \mathcal{P} \longrightarrow 2^W$, $\nu : \mathcal{N} \longrightarrow W$

$$
\begin{array}{lll}
\mathcal{M}, x \models p & \text{iff} & x \in v(p) \\
\mathcal{M}, x \models @_j \phi & \text{iff} & \mathcal{M}, v_0(j) \models \phi \\
\mathcal{M}, x \not\models \bot & & \\
\mathcal{M}, x \models \neg \phi & \text{iff} & \mathcal{M}, x \not\models \phi \\
\mathcal{M}, x \models \phi \vee \psi & \text{iff} & \mathcal{M}, x \models \phi \text{ or } \mathcal{M}, x \models \psi \\
\mathcal{M}, x \models \Box \phi & \text{iff} & \text{for all } R\text{-successors } y \text{ of } x,\ \mathcal{M}, y \models \phi
\end{array}
$$

R is a pre-order (reflexive and transitive).

Specification Languages of the Framework

- Object specification language for specifying syntax of a logic.
- Semantic specification language for specifying semantics of a logic which syntax is specified in the object language.

Object Specification Language of the Framework

$\mathcal{L}$ = propositional many-sorted language:

- $\{0, 1, \ldots, N\}$ = an index set of sorts. 1 = primary sort
- A countable set of the logical connectives.
- Sort of a m-ary connective ($m \geq 0$):

$$(i_1, \ldots, i_m, i) \in \{0, 1, \ldots, N\}^{m+1}.$$

Notation: $(i_1, \ldots, i_m) \mapsto i$.

- Agreement: 0-ary connective of a sort $\mapsto i$ = constant of the sort i.
- $\mathcal{L}^i$ = expressions of the sort $i = 0, \ldots, N$:
  - a countable set of variables $p_0^i, p_1^i, \ldots$;
  - For every connective σ of a sort $(i_1, \ldots, i_m) \mapsto i$, $\sigma(E_1, \ldots, E_m) \in \mathcal{L}^i$ whenever $E_1 \in \mathcal{L}^{i_1}, \ldots, E_m \in \mathcal{L}^{i_m}$.
- $\mathcal{L} \stackrel{\text{def}}{=} \bigcup_{i \in Sorts} \mathcal{L}^i$.

Examples

Modal Logic (K, S4, etc)

formulae = expressions of the sort 1

connectives :

sort 1: $\bot$

sort $1 \mapsto 1$: $\neg$, $\Box$

sort $(1, 1) \mapsto 1$: $\vee$

Dynamic Modal Logic

formulae = expressions of the sort 1

relations = expressions of the sort 2

connectives :

sort 1: $\bot$

sort $1 \mapsto 1$: $\neg$

sort $(1, 1) \mapsto 1$: $\vee$

sort $(2, 1) \mapsto 1$: $[\cdot]$

Language for Hybrid Logic S4$(@)$

Connectives:

$\neg$ and $\Box$ of the sort $1 \mapsto 1$,

$\vee$ of the sort $(1, 1) \mapsto 1$,

$\bot$ of the sort 1,

$@$ of the sort $(0, 1) \mapsto 1$.

Formulae = expressions of the sort 1
Nominals = expressions of the sort 0

Semantic Specification Language = Many-Sorted First-Order Language

$FO(\mathcal{L})$ = an extension of $\mathcal{L}$ with

- a sort $N+1$ which is called *domain sort*:
  - variables $x, y, z, \ldots$ of the sort $N+1$
  - constants $a, b, c, \ldots$ of the sort $N+1$
  - predicate symbols $P, Q, R \ldots$ on the sort $N+1$
  - function symbols $f, g, h, \ldots$ which act on the sort $N+1$
- the equality predicate symbol $\approx$
- symbols $\nu_0, \ldots, \nu_N$ = "interpretation" symbols:
  - ν_0 = a function symbol from the sort 0 to the sort $N+1$
  - ν_n $(n > 0)$ = "holds" predicate,
    a predicate symbol which accepts arguments $(E, t_1, \ldots, t_n)$
    for an expression E of the sort n
    and terms $t_1, \ldots, t_n$ of the sort $N+1$.

$FO(\mathcal{L})$ is a many-sorted first-order language where connectives of $\mathcal{L}$ are transformed into functional symbols of appropriate sorts with added "interpretation" symbols $\nu_0, \ldots, \nu_N$.

Example: Semantic Specification Language for S4$(@)$

- $N = 1$.
- The sort 0 = nominals.
- The sort 1 = formula terms.
- The sort 2 = domain sort ($N + 1 = 2$).
- The functional symbols obtained from the connectives of S4$(@)$:
  - a constant $\bot$ (of the sort 1),
  - unary functional symbols $\neg$ and $\Box$ of the sort $1 \mapsto 1$,
  - binary functional symbol $\vee$ of the sort $(1, 1) \mapsto 1$,
  - binary functional symbol $@$ of the sort $(0, 1) \mapsto 1$,
- A constant binary predicate symbol R (of the sort $(2, 2)$).
- The equality symbol $\approx$.
- The "interpretation" symbols ν_0, ν_1:
  - for every nominal i,
    $\nu_0(i)$ is a term of the sort 2,
  - for every S4$(@)$ formula ϕ and a term t of the sort 2,
    $\nu_1(\phi, t)$ is an atomic formula of *FO*(S4$(@)$).

Notation

- A sequence of first-order variables: $\overline{w} \stackrel{\text{def}}{=} w_1, \ldots, w_n$ for some $n > 0$.
- Universal quantifier prefix: $\forall\overline{w} \stackrel{\text{def}}{=} \forall w_1 \cdots \forall w_n$.
- $\phi(\overline{w})$ indicates that all the free variables of the first-order formula ϕ are belong to the set $\{w_1, \ldots, w_n\}$.
- *Universal closure* of a set of formulae S:

$$\forall S \stackrel{\text{def}}{=} \{\forall\overline{w}\ \phi(\overline{w}) \mid \phi(\overline{w}) \in S\}.$$

$\mathcal{L}$-Open Formulae = Unbounded Formulae wrt $\mathcal{L}$

- Formulae of $FO(\mathcal{L})$ in which all occurrences of the $\mathcal{L}$-variables of sorts $i = 0, \ldots, N$ are free are called $\mathcal{L}$-*open* formulae.
- An $\mathcal{L}$-*open sentence* is an $\mathcal{L}$-open formula that does *not* have free occurrences of variables of the domain sort $N + 1$.

Example 5

- $\mathcal{L}_{\mathsf{S4}(@)}$-open formula but *not* $\mathcal{L}_{\mathsf{S4}(@)}$-open sentence:

$$\exists y\,(\nu_1(\Box p, y) \wedge R(x, y))$$

- $\mathcal{L}_{\mathsf{S4}(@)}$-open sentence:

$$\exists y\,(\nu_1(\Box p, y) \wedge \forall x\, R(x, y))$$

- *Not* $\mathcal{L}_{\mathsf{S4}(@)}$-open formulae:

$$\forall p \forall y\,(\nu_1(\Box p, y) \wedge R(x, y)) \quad \text{and} \quad \exists i\,(\nu_1(@_i p, y) \wedge \nu_0(i) \approx y))$$

Connective Definitions

Let S be any set of $\mathcal{L}$-open sentences in $FO(\mathcal{L})$ and σ be a connective.

- A $FO(\mathcal{L})$- formula ϕ^{σ} *defines a connective* σ with respect to a set of $\mathcal{L}$-open sentences S if it does not contain σ and the following holds:

$$\forall S \models \forall \overline{p}\ \forall \overline{x}\ (\nu_n(\sigma(\overline{p}), \overline{x}) \equiv \phi^{\sigma}(\overline{p}, \overline{x})).$$

- *σ-definition with respect to S*:

$$\forall \overline{x}\ (\nu_n(\sigma(\overline{p}), \overline{x}) \equiv \phi^{\sigma}(\overline{p}, \overline{x})),$$

Example: $\Box$ Connective

$\Box$-definition:

$$\forall x\ \Big(\nu_1(\Box p, x) \equiv \underbrace{\forall y\ (R(x,y) \rightarrow \nu_1(p,y))}_{\phi^{\Box}}\Big).$$

Semantic Specification

Definition 6

A *(first-order) semantic specification* of the object language $\mathcal{L}$ is a set S of $\mathcal{L}$-open $FO(\mathcal{L})$-sentences defining all the connectives of $\mathcal{L}$.

We assume that any semantic specification includes the equality axioms.

Equality axioms

$$\forall x\ (x \approx x) \qquad \forall x \forall y\ (x \approx y \rightarrow y \approx x) \qquad \forall x \forall y \forall z\ (x \approx y \wedge y \approx z \rightarrow x \approx z)$$

$$\forall \overline{x} \forall y_i\ \ (P(\overline{x}) \wedge x_i \approx y_i \rightarrow P(x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n))$$

$$\forall p \forall \overline{x} \forall y_i\ \ (\nu_n(p, \overline{x}) \wedge x_i \approx y_i \rightarrow \nu_n(p, x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n))$$

$$\forall \overline{p} \forall \overline{x} \forall y_i\ (x_i \approx y_i \rightarrow f(\overline{p}, \overline{x}) \approx f(\overline{p}, x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n))$$

Every semantic specification S is associated with its set of connective definitions S^0.

A Very Common Semantic Specification

- $S = S^0 \cup S^b$.
- S^0 = connective definitions (for every connective σ):

$$\forall \overline{x} \ (\nu_n(\sigma(\overline{p}), \overline{x}) \equiv \phi^{\sigma}(\overline{p}, \overline{x})).$$

- S^b = background theory: no non-atomic $\mathcal{L}$-expressions.
- S^b contains the usual equality axioms.

Example: Semantic Specification of S4(@)

Connective definitions in $S^0_{\mathsf{S4}(@)}$

$$\forall x\ \big(\nu_1(\bot, x) \equiv \bot\big)$$
$$\forall x\ \big(\nu_1(\neg p, x) \equiv \neg\nu_1(p, x)\big)$$
$$\forall x\ \big(\nu_1(p \vee q, x) \equiv \nu_1(p, x) \vee \nu_1(q, x)\big)$$
$$\forall x\ \big(\nu_1(\Box p, x) \equiv \forall y\ \big(R(x, y) \rightarrow \nu_1(p, y)\big)\big)$$
$$\forall x\ \big(\nu_1(@_i p, x) \equiv \nu_1(p, \nu_0(i))\big)$$

Reflexivity and transitivity axioms in $S^b_{\mathsf{S4}(@)}$

$$\forall x\ R(x, x)$$
$$\forall x \forall y \forall z\ \big((R(x, y) \wedge R(y, z)) \rightarrow R(x, z)\big)$$

+the equality axioms for $FO(\mathsf{S4}(@))$

Normalised Semantic Specification

Normalisation is required to reduce sentences in a semantic specification to implicative forms which are convenient for generating tableau rules.

Definition 7

S is *normalised* iff $S = S^+ \cup S^- \cup S^b$, where S^+, S^- and S^b are disjoint sets of sentences satisfying the following:

- S^+ is a set of $\mathcal{L}$-open sentences of the form:

$$\xi^E_+ \stackrel{\text{def}}{=} \forall \overline{x}\ (\nu_n(E(p_1, \ldots, p_m), \overline{x}) \rightarrow \phi^E_+(p_1, \ldots, p_m, \overline{x})).$$

- S^- is a set of $\mathcal{L}$-open sentences of the form:

$$\xi^E_- \stackrel{\text{def}}{=} \forall \overline{x}\ (\phi^E_-(p_1, \ldots, p_m, \overline{x}) \rightarrow \nu_n(E(p_1, \ldots, p_m), \overline{x})).$$

- All $\mathcal{L}$-expressions occurring in S^b are atomic.

Note that E can be any expression, not necessarily of the form $\sigma(\overline{p})$.

Every normalised semantic specification S includes a background theory S^b but may not coincide with $S^0 \cup S^b$.

Example: Normalising $S_{\mathsf{S4}(@)}$

Decomposing the connective definitions in $S^{0}_{\mathsf{S4}(@)}$ into $S^{+}_{\mathsf{S4}(@)}$-sentences and $S^{-}_{\mathsf{S4}(@)}$-sentences as follows.

$S^{+}_{\mathsf{S4}(@)}$-sentences:

$$\begin{aligned}
\forall x\ \big(\nu_1(\bot, x) &\rightarrow \bot\big)\\
\forall x\ \big(\nu_1(\neg p, x) &\rightarrow \neg\nu_1(p, x)\big)\\
\forall x\ \big(\nu_1(p \vee q, x) &\rightarrow \nu_1(p, x) \vee \nu_1(q, x)\big)\\
\forall x\ \big(\nu_1(\Box p, x) &\rightarrow \forall y\ \big(R(x, y) \rightarrow \nu_1(p, y)\big)\big)\\
\forall x\ \big(\nu_1(@_i p, x) &\rightarrow \nu_1(p, \nu_0(i))\big)
\end{aligned}$$

$S^{-}_{\mathsf{S4}(@)}$-sentences:

$$\begin{aligned}
\forall x\ \big(\bot &\rightarrow \nu_1(\bot, x)\big)\\
\forall x\ \big(\neg\nu_1(p, x) &\rightarrow \nu_1(\neg p, x)\big)\\
\forall x\ \big(\nu_1(p, x) \vee \nu_1(q, x) &\rightarrow \nu_1(p \vee q, x)\big)\\
\forall x\ \big(\forall y\ \big(R(x, y) \rightarrow \nu_1(p, y)\big) &\rightarrow \nu_1(\Box p, x)\big)\\
\forall x\ \big(\nu_1(p, \nu_0(i)) &\rightarrow \nu_1(@_i p, x)\big)
\end{aligned}$$

Induced Expression Ordering

Well-founded expression ordering will be needed for the termination proof.

Definition 8

For a normalised semantic specification S, *S-induced relation* $\prec$ is the smallest transitive relation on $\mathcal{L}$-expressions such that for all expressions E' and E,
$E' \prec E$ whenever for some $\mathcal{L}$-substitution θ and for some sentence ξ^F_+ (ξ^F_-) in S,

- $E = F\theta$ and
- E' occurs in $\phi^F_+\theta$ ($\phi^F_-\theta$, respectively).

The reflexive closure of $\prec$ is denoted by $\preceq$.

- In general, $\prec$ is not an ordering.
- Usually $\prec$ is well-founded ordering

  (because of the standard inductive way of defining interpretations).

Example: Subexpression Ordering for S4(@)

$\prec$ is the smallest transitive ordering which satisfies for all nominals i and formulae ϕ and ψ:

- $\phi \prec \neg\phi$ $\quad [\forall x\ (\nu_1(\neg p, x) \rightarrow \neg\nu_1(p, x))]$
- $\phi \prec \phi \vee \psi$ $\quad [\forall x\ (\nu_1(p \vee q, x) \rightarrow (\nu_1(p, x) \vee \nu_1(q, x))]$
- $\psi \prec \phi \vee \psi$
- $\phi \prec \Box\phi$ $\quad [\forall x\ (\nu_1(\Box p, x) \rightarrow \forall y\ (R(x, y) \rightarrow \nu_1(p, y)))]$
- $i \prec @_i\phi$ $\quad [\forall x\ (\nu_1(@_i p, x) \rightarrow \nu_1(p, \nu_0(i)))]$
- $\phi \prec @_i\phi$

$\prec$ is a subexpression ordering and, hence, it is well-founded!

Subexpression Operator

- $\mathrm{sub}_{\prec}(E) \stackrel{\text{def}}{=} \{E' \mid E' \prec E\}$ and $\mathrm{sub}_{\prec}(X) \stackrel{\text{def}}{=} \bigcup_{E \in X} \mathrm{sub}_{\prec}(E)$.
- $\mathrm{sub}_{\prec}(X)$ is the set of all expressions $\prec$-smaller than some expression in X.
- An operator sub (which maps sets of expressions to sets of expressions) is *finite* if $\mathrm{sub}(X)$ is finite for every finite set of expressions X.

Consequence of König's Infinity Lemma

If $\prec$ is finitely branching and well-founded ordering then $\mathrm{sub}_{\prec}$ is finite.

- If, in normalised semantic specification S, the set $S^+ \cup S^-$ is finite then $\prec$ is finitely branching.
- If, in normalised semantic specification S, all ϕ_+^E and ϕ_-^E contain only atomic $\mathcal{L}$-expressions then $\prec$ is well-founded.

Notation

- $\Phi^{\boldsymbol{E}}_{+} \stackrel{\text{def}}{=} \{\phi^{\boldsymbol{F}}_{+}\theta \mid \boldsymbol{E} = \boldsymbol{F}\theta \text{ for some } \xi^{\boldsymbol{F}}_{+} \text{ from } \boldsymbol{S}\}$,
- $\Phi^{\boldsymbol{E}}_{-} \stackrel{\text{def}}{=} \{\phi^{\boldsymbol{F}}_{-}\theta \mid \boldsymbol{E} = \boldsymbol{F}\theta \text{ for some } \xi^{\boldsymbol{F}}_{-} \text{ from } \boldsymbol{S}\}$.

Example 9

$$\Phi^{\Box(p \vee q)}_{+} = \Phi^{\Box(p \vee q)}_{-} = \{\forall y\,(\boldsymbol{R}(x,y) \rightarrow \nu_1(p \vee q, y))\}.$$

Notation

$$S\restriction X \stackrel{\text{def}}{=} \{\phi\theta \mid \phi \in S \text{ , } \theta \text{ is } \mathcal{L}\text{-substitution and all } \mathcal{L}\text{-expressions occurring in } \phi\theta \text{ belong to } X\},$$

Example 10

$$S \stackrel{\text{def}}{=} \{\nu_1(@_i p, y), \nu_1(\neg p, x)\} \quad \text{and} \quad X \stackrel{\text{def}}{=} \{i_0, p_0, p, p \vee p_0, @_{i_0} p_0\}.$$

Instantiations of formulae in S relative to X:

$$\nu_1(@_{i_0} p_0, x),\ \nu_1(@_{i_0} p, x),\ \nu_1(@_{i_0}(p \vee p_0), x),\ \nu_1(@_{i_0} @_{i_0} p_0, x),$$
$$\nu_1(\neg p_0, x),\ \nu_1(\neg p, x),\ \nu_1(\neg(p \vee p_0), x),\ \nu_1(\neg @_{i_0} p_0, x).$$

$$S\restriction X = \{\nu_1(@_{i_0} p_0, y)\}.$$

Well-Defined Semantic Specification

Well-defined semantic specification will allow to generate a sound and complete tableau calculus.

Definition 11 (Well-definedness)

A semantic specification S is *well-defined* iff S is normalised and the following conditions are all true.

1. $\forall S^0, \forall S^b \models \forall S$,
2. the relation $\prec$ induced by S is a well-founded ordering, and
3. for every expression $E = \sigma(\overline{p})\theta$,
   $\forall S^0, S^b \restriction \mathsf{sub}_\prec(E) \models \forall \overline{x}\Big(\Big(\bigwedge \Phi^E_+ \rightarrow \phi^\sigma\theta\Big) \wedge \Big(\phi^\sigma\theta \rightarrow \bigvee \Phi^E_-\Big)\Big)$.

- It is enough to check the third condition for such E that $\xi^E_+ \in S$ or $\xi^E_- \in S$.
- If S is finite and there are only finite number of connectives, the third condition can be rewritten into a first-order formula!

Example

- Since $S_{\mathsf{S4}(@)} = S^0_{\mathsf{S4}(@)} \cup S^b_{\mathsf{S4}(@)}$,

$$\forall S^0_{\mathsf{S4}(@)}, \forall S^b_{\mathsf{S4}(@)} \models \forall S_{\mathsf{S4}(@)}.$$

- $\prec$ (induced by $S_{\mathsf{S4}(@)}$) is the direct subexpression ordering and, hence, well-founded.
- Recall that $\Phi^{\Box p}_{+} = \Phi^{\Box p}_{-} = \{\forall y\,(R(x,y) \rightarrow \nu_1(p,y))\}$, and

$$\phi^{\Box}(p,x) = \forall y\,(R(x,y) \rightarrow \nu_1(p,y)).$$

- The formula

$$\forall \overline{x}\Big(\Big(\bigwedge \Phi^{\Box p}_{+} \rightarrow \phi^{\Box}(p,x)\Big) \wedge \Big(\phi^{\Box}(p,x) \rightarrow \bigvee \Phi^{\Box p}_{-}\Big)\Big)$$

  is a tautology.
- Similarly for other connectives.

Any semantic specification S in the form $S = S^0 \cup S^b$ is usually well-defined!

Model

A $FO(\mathcal{L})$-model is a usual first-order model for the many-sorted language $FO(\mathcal{L})$:

$$\mathcal{I} \stackrel{\text{def}}{=} (\Delta_0^{\mathcal{I}}, \ldots, \Delta_{N+1}^{\mathcal{I}}, P^{\mathcal{I}}, \ldots, f^{\mathcal{I}}, \ldots, \sigma^{\mathcal{I}}, \ldots, \nu_0^{\mathcal{I}}, \ldots, \nu_N^{\mathcal{I}})$$

where

- $\Delta_0^{\mathcal{I}}, \ldots, \Delta_N^{\mathcal{I}}, \Delta_{N+1}^{\mathcal{I}}$ are non-empty sets,
- $P^{\mathcal{I}} \subseteq (\Delta_{N+1}^{\mathcal{I}})^m$, where m is the arity of P,
- $f^{\mathcal{I}} : (\Delta_{N+1}^{\mathcal{I}})^m \longrightarrow \Delta_{N+1}^{\mathcal{I}}$ for every m-ary function symbol f,
- $\sigma^{\mathcal{I}} : \Delta_{i_1}^{\mathcal{I}} \times \cdots \times \Delta_{i_m}^{\mathcal{I}} \longrightarrow \Delta_i^{\mathcal{I}}$ for every connective σ of a sort $(i_1, \ldots, i_m) \mapsto i$,
- $\nu_0(\ell)^{\mathcal{I}} \in \Delta_{N+1}^{\mathcal{I}}$ for every $\ell \in \mathcal{L}^0$,
- $\nu_n^{\mathcal{I}} \subseteq \Delta_n^{\mathcal{I}} \times (\Delta_{N+1}^{\mathcal{I}})^n$, for $0 < n \leq N$.

Standard Notions of Valuation and Truth in a Model

- A *valuation* in a model $\mathcal{I}$ is a mapping ι from the set of variables and constants of each sort $n = 0, \ldots, N+1$ of $FO(\mathcal{L})$ to the corresponding $\Delta_n^{\mathcal{I}}$.
- Lift valuation to all terms:

$$\iota f(t_1, \ldots, t_n) \stackrel{\text{def}}{=} f^{\mathcal{I}}(\iota t_1, \ldots, \iota t_n)$$

  for any functional symbol f (including symbols turned from connectives).

- For any predicate symbol p (including ν_n)

$$\mathcal{I}, \iota \models p(t_1, \ldots, t_n) \text{ iff } (\iota t_1, \ldots, \iota t_n) \in p^{\mathcal{I}}.$$

- Lift interpretation to all formulae:
  - $\mathcal{I}, \iota \models \neg\phi$ iff $\mathcal{I}, \iota \not\models \phi$
  - $\mathcal{I}, \iota \models \phi \vee \psi$ iff $\mathcal{I}, \iota \models \phi$ or $\mathcal{I}, \iota \models \psi$
  - $\mathcal{I}, \iota \models \forall x \phi$ iff $\mathcal{I}, \varkappa \models \phi$ for every valuation $\varkappa$ such that $\iota y = \varkappa y$ for every variable $y \neq x$.
- Notation: $\mathcal{I}, \iota \models \mathcal{S}$ iff $\mathcal{I}, \iota \models \phi$ for every formula ϕ in $\mathcal{S}$.

$\mathcal{L}$-Structure

- Recall that $\mathcal{L}^i$ is the set of logical expressions of the sort i.
- Because $\mathcal{L}$-connectives become functional symbols of $FO(\mathcal{L})$, $\mathcal{L}^i$ is also the set of all terms of the sort i in the language $FO(\mathcal{L})$.
- An *$\mathcal{L}$-structure* is a $FO(\mathcal{L})$-model which has the shape

$$\mathcal{I} \stackrel{\text{def}}{=} (\mathcal{L}^0, \ldots, \mathcal{L}^N, \Delta^{\mathcal{I}}, P^{\mathcal{I}}, \ldots, f^{\mathcal{I}}, \ldots, \sigma^{\mathcal{I}}, \ldots, \nu_0^{\mathcal{I}}, \ldots, \nu_N^{\mathcal{I}}).$$

- Simplified notation:

$$\mathcal{I} \stackrel{\text{def}}{=} (\Delta^{\mathcal{I}}, P^{\mathcal{I}}, \ldots, f^{\mathcal{I}}, \ldots, \nu_0^{\mathcal{I}}, \ldots, \nu_N^{\mathcal{I}}).$$

- Even simpler:

$$\mathcal{I} \stackrel{\text{def}}{=} (\Delta^{\mathcal{I}}, P^{\mathcal{I}}, \ldots, f^{\mathcal{I}}, \ldots, \nu^{\mathcal{I}}).$$

- A very similar notation is used for models of modal and description logics!
- *$\mathcal{L}$-frame*: $\mathcal{F} \stackrel{\text{def}}{=} (\Delta^{\mathcal{I}}, P^{\mathcal{I}}, \ldots, f^{\mathcal{I}}, \ldots)$.

- $\mathcal{L}$-structure is a special $FO(\mathcal{L})$-model.
- $\mathcal{L}$-structure serves as a model for the language $\mathcal{L}$.

Canonical Valuation in $\mathcal{L}$-Structure

- A valuation ι in an $\mathcal{L}$-structure is *canonical* if every variable and constant of any sort $i = 0, \ldots, N$ is mapped to itself.
- A canonical valuation maps any term of any sort $i = 0, \ldots, N$ to the term itself.

Main Theorem about $\mathcal{L}$-Open Formulae

Theorem 12

For every FO($\mathcal{L}$)-model $\mathcal{J}$ and a valuation ι there is an $\mathcal{L}$-structure $\mathcal{I}$ and a canonical valuation $\varkappa$ such that for any $\mathcal{L}$-open formula ϕ

$$\mathcal{J}, \iota \models \phi \text{ iff } \mathcal{I}, \varkappa \models \phi.$$

Classes of FO($\mathcal{L}$)-models and $\mathcal{L}$-structures cannot be distinguished by an $\mathcal{L}$-open formula.

- We consider only $\mathcal{L}$-open formulae.
- We do not care about canonicity of valuations.

First-Order Definable Logic

- An exact meaning of first-order definability for propositional logics:

Definition 13 (First-Order Definability)

A logic L in the language $\mathcal{L}$ is *first-order definable* iff there is a semantic specification S_L in the language $FO(\mathcal{L})$ such that

$$L = \{E \in \mathcal{L}^n \mid \forall S_L \models \forall \overline{x}\ \nu_n(E, \overline{x})\} \text{ for some } n = 0, \ldots, N.$$

- In the case of modal logics:

$$L = \{A \in \mathcal{L}^1 \mid \forall S_L \models \forall x\ \nu_1(A, x)\}.$$

Well-Known Facts

- Every semantic specification defines a class of $FO(\mathcal{L})$-models (or, equivalently, $\mathcal{L}$-structures) on which it is true.
- There are classes of $FO(\mathcal{L})$-models for which there are no semantic specifications.
- In general, a semantic specification for given class of $FO(\mathcal{L})$-models is not unique.

Session Summary

- We introduced two specification languages:
  - $\mathcal{L}$ — object language for specification of syntax of a logic and
  - $FO(\mathcal{L})$ — first-order language for specification of semantics of a logic.
- We defined the notion of semantic specification.
- We formalised the notion of “first-order definable logic”.
- We introduced the notions of normalised and well-defined semantic specification.
- We demonstrated the new definitions on the example of S4$(@)$ and saw that many standard logics have well-defined semantic specification.

Our running example S4(@): The standard definition

- Nominals: i
  Formulae: $\phi, \psi \longrightarrow p \mid \bot \mid \neg\phi \mid \phi \vee \psi \mid \Box\phi \mid @_i\phi$

- Semantics: Kripke model $\mathcal{M} = (W, R, v, v_0)$

  valuation mapping $v : \mathcal{P} \longrightarrow 2^W$

  valuation mapping $v_0 : \mathcal{N} \longrightarrow W$

$$
\begin{array}{lll}
\mathcal{M}, x \models p & \text{iff} & x \in v(p) \\
\mathcal{M}, x \not\models \bot & & \\
\mathcal{M}, x \models \neg\phi & \text{iff} & \mathcal{M}, x \not\models \phi \\
\mathcal{M}, x \models \phi \vee \psi & \text{iff} & \mathcal{M}, x \models \phi \text{ or } \mathcal{M}, x \models \psi \\
\mathcal{M}, x \models \Box\phi & \text{iff} & \text{for all } R\text{-successors } y \text{ of } x \ \mathcal{M}, y \models \phi \\
\mathcal{M}, x \models @_i\phi & \text{iff} & \mathcal{M}, v_0(i) \models \phi
\end{array}
$$

- R is reflexive and transitive

Specification of S4(@) in the framework

- S^0: Definitions of the semantics of the connectives

$$\begin{aligned}
\forall x \left[\nu_1(\bot, x) \equiv \bot \right] \\
\forall x \left[\nu_1(\neg p, x) \equiv \neg \nu_1(p, x) \right] \\
\forall x \left[\nu_1(p \vee q, x) \equiv \nu_1(p, x) \vee \nu_1(q, x) \right] \\
\forall x \left[\nu_1(\Box p, x) \equiv \forall y \left(R(x, y) \rightarrow \nu_1(p, y) \right) \right] \\
\forall x \left[\nu_1(@_i p, x) \equiv \nu_1(p, \nu_0(i)) \right]
\end{aligned}$$

- S^b: Background theory specifying the frame conditions of R

$$\begin{aligned}
\forall x \, R(x, x) \\
\forall x \, \forall y \, \forall z \, (R(x, y) \wedge R(y, z) \rightarrow R(x, z))
\end{aligned}$$

Extracting tableau rules

- Definition of semantics of connective σ:

$$\forall \overline{x} \left[\nu_n(\sigma(\overline{p}), \overline{x}) \equiv \phi(\overline{p}, \overline{x}) \right]$$

- Take the *left-to-right implication* of the definition of a connective

$$\forall \overline{x} \left[\nu_n(\sigma(\overline{p}), \overline{x}) \rightarrow \phi(\overline{p}, \overline{x}) \right]$$

- Eliminate quantifiers, Skolemise if needed, and rewrite into this form

$$\nu_n(\sigma(\overline{p}), \overline{x}) \rightarrow \bigvee_{j=1}^{J} \bigwedge_{k=1}^{K_j} \psi_{jk}$$

- Rewrite as this rule:

$$\frac{\nu_n(\sigma(\overline{p}), \overline{x})}{\psi_{11}, \ \ldots, \ \psi_{1K_1} \mid \cdots \mid \psi_{J1}, \ \ldots, \ \psi_{JK_J}}$$

ψ_{jk} = literal

Extracting tableau rules (cont'd)

- Do the same for the *right-to-left implication*:
  Take the contra-positive (!), eliminate quantifiers, Skolemise if needed, and rewrite to implicational DNF and as rule:

$$\frac{\neg\nu_n(\sigma(\overline{p}),\overline{x})}{\psi_{11},\ \ldots,\ \psi_{1K_1} \mid \cdots \mid \psi_{J1},\ \ldots,\ \psi_{JK_J}}$$

- For each formula in the background theory S^b:
  Eliminate quantifiers, Skolemise if needed, and transform into DNF; then turn into one rule:

$$\frac{}{\psi_{11},\ \ldots,\ \psi_{1K_1} \mid \cdots \mid \psi_{J1},\ \ldots,\ \psi_{JK_J}}$$

Extracting rules for S4(@)

- Conversion for left-to-right definition of $\Box$:

$$\forall x \left[\nu_1(\Box p, x) \rightarrow \forall y \left(R(x,y) \rightarrow \nu_1(p,y) \right) \right]$$

$$\nu_1(\Box p, x) \rightarrow .\neg R(x,y) \vee \nu_1(p,y)$$

$$\frac{\nu_1(\Box p, x)}{\neg R(x,y) \mid \nu_1(p,y)}$$

- Conversion for right-to-left definition of $\Box$:

$$\forall x \left[\nu_1(\Box p, x) \leftarrow \forall y \left(R(x,y) \rightarrow \nu_1(p,y) \right) \right]$$

$$\forall x \left[\neg\nu_1(\Box p, x) \rightarrow \neg\forall y \left(R(x,y) \rightarrow \nu_1(p,y) \right) \right]$$

$$\neg\nu_1(\Box p, x) \rightarrow .R(x, f(p,x)) \wedge \neg\nu_1(p, f(p,x))$$

$$\frac{\neg\nu_1(\Box p, x)}{R(x, f(p,x)),\ \ \neg\nu_1(p, f(p,x))}$$

f is fresh $\Longrightarrow$ automatic way to force new concrete element to be introduced

Extracting rules for S4(@) (cont'd)

- Conversion of the frame properties in the background theory: Reflexivity

$$\forall x\, R(x,x)$$

$$R(x,x)$$

$$\overline{R(x,x)}$$

- Conversion of the frame properties in the background theory: Transitivity

$$\forall x\, \forall y\, \forall z\ \big(R(x,y) \wedge R(y,z) \rightarrow R(x,z)\big)$$

$$\neg R(x,y) \vee \neg R(y,z) \vee R(x,z)$$

$$\overline{\neg R(x,y) \mid \neg R(y,z) \mid R(x,z)}$$

Closure rules

- In general, include:

$$\frac{\nu_n(p,\overline{x}),\ \ \neg\nu_n(p,\overline{x})}{\bot}$$

for each sort $n \geq 1$

$$\frac{P(\overline{x}),\ \ \neg P(\overline{x})}{\bot}$$

for each constant predicate symbol P

- For S4(@), include:

$$\frac{\nu_1(p,x),\ \ \neg\nu_1(p,x)}{\bot}$$

$$\frac{R(x,y),\ \ \neg R(x,y)}{\bot}$$

Default equality rules

- Include (for each sort n, constant predicate symbol P, Skolem function f):

$\approx$ is an equivalence:

$$\frac{}{x \approx x} \qquad \frac{x \approx y}{y \approx x} \qquad \frac{x \approx y,\ \ y \approx z}{x \approx z}$$

Predicate substitutivity:

$$\frac{x \approx y,\ \ \nu_n(p, x_1, \ldots x_{i-1}, x, x_{i+1}, \ldots, x_n)}{\nu_n(p, x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n)} \qquad n \geq 1$$

$$\frac{x \approx y,\ \ P(x_1, \ldots x_{i-1}, x, x_{i+1}, \ldots, x_n)}{P(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_m)}$$

Functional substitutivity:

$$\frac{i \approx j}{\nu_0(i) \approx \nu_0(j)}$$

$$\frac{x \approx y}{f(\overline{p}, x_1, \ldots x_{i-1}, x, x_{i+1}, \ldots, x_m) \approx f(\overline{p}, x_1, \ldots x_{i-1}, y, x_{i+1}, \ldots, x_m)}$$

- Remark: These rules can be obtained using rule synthesis (and rule refinement, see later) from the standard equality axioms

Default equality rules for S4(@)

$\approx$ is an equivalence:

$$\frac{}{x \approx x} \qquad \frac{x \approx y}{y \approx x} \qquad \frac{x \approx y,\ \ y \approx z}{x \approx z}$$

Predicate substitutivity:

$$\frac{x \approx y,\ \ \nu_1(p,x)}{\nu_1(p,y)}$$

$$\frac{x \approx y,\ \ R(x,x_2)}{R(y,x_2)} \qquad \frac{x \approx y,\ \ R(x_1,x)}{R(x_1,y)}$$

Functional substitutivity:

$$\frac{i \approx j}{\nu_0(i) \approx \nu_0(j)}$$

$$\frac{x \approx y}{f(p,x) \approx f(p,y)}$$

Generated tableau calculus

Generated tableau calculus Tab_S consists of:

- *Decomposition rules* generated from definitions in S^0
- *Theory rules* generated from the background theory S^b
- Closure rules
- Default equality rules (if $\approx$ was used in S and/or blocking is required)

Using a generated tableau calculus:

- Is ϕ satisfiable wrt. S ?

  $\Longrightarrow$ Construct a tableau derivation for $\nu_1(\phi, a)$

Note: ϕ is a formula of the primary sort, by assumption sort 1
a = Skolem constant for $\exists x$ in $\exists x\ \nu_1(\phi, x)$

- Is ϕ satisfiable wrt. $S_{\mathsf{S4}(@)}$?

  $\Longrightarrow$ Start the tableau derivation with $\nu_1(\phi, a)$

The generated tableau calculus for S4(@)

- Decomposition rules

$$\frac{\nu_1(\bot, x)}{\bot} \qquad \frac{\neg\nu_1(\bot, x)}{\neg\bot} \qquad \frac{\nu_1(\neg p, x)}{\neg\nu_1(p, x)} \qquad \frac{\neg\nu_1(\neg p, x)}{\nu_1(p, x)}$$

$$\frac{\nu_1(p \vee q, x)}{\nu_1(p, x) \mid \nu_1(q, x)} \qquad \frac{\neg\nu_1(p \vee q, x)}{\neg\nu_1(p, x),\ \neg\nu_1(q, x)}$$

$$\frac{\nu_1(\Box p, x)}{\neg R(x, y) \mid \nu_1(p, y)} \qquad \frac{\neg\nu_1(\Box p, x)}{R(x, f(p, x)),\ \neg\nu_1(p, f(p, x))}$$

$$\frac{\nu_1(@_i p, x)}{\nu_1(p, \nu_0(i))} \qquad \frac{\neg\nu_1(@_i p, x)}{\neg\nu_1(p, \nu_0(i))}$$

- Theory rules

$$\frac{}{R(x, x)} \qquad \frac{}{\neg R(x, y) \mid \neg R(y, z) \mid R(x, z)}$$

- Closure rules

$$\frac{\nu_1(p, x),\ \neg\nu_1(p, x)}{\bot} \qquad \frac{R(x, y),\ \neg R(x, y)}{\bot}$$

Rule application

- Variables p and q can match with formulae (sort 1)
  Variables x, y and z can match with terms of domain sort $N+1$
  Variable i can match with nominals (sort 0)
  R and f are external constant symbols which match only with themselves

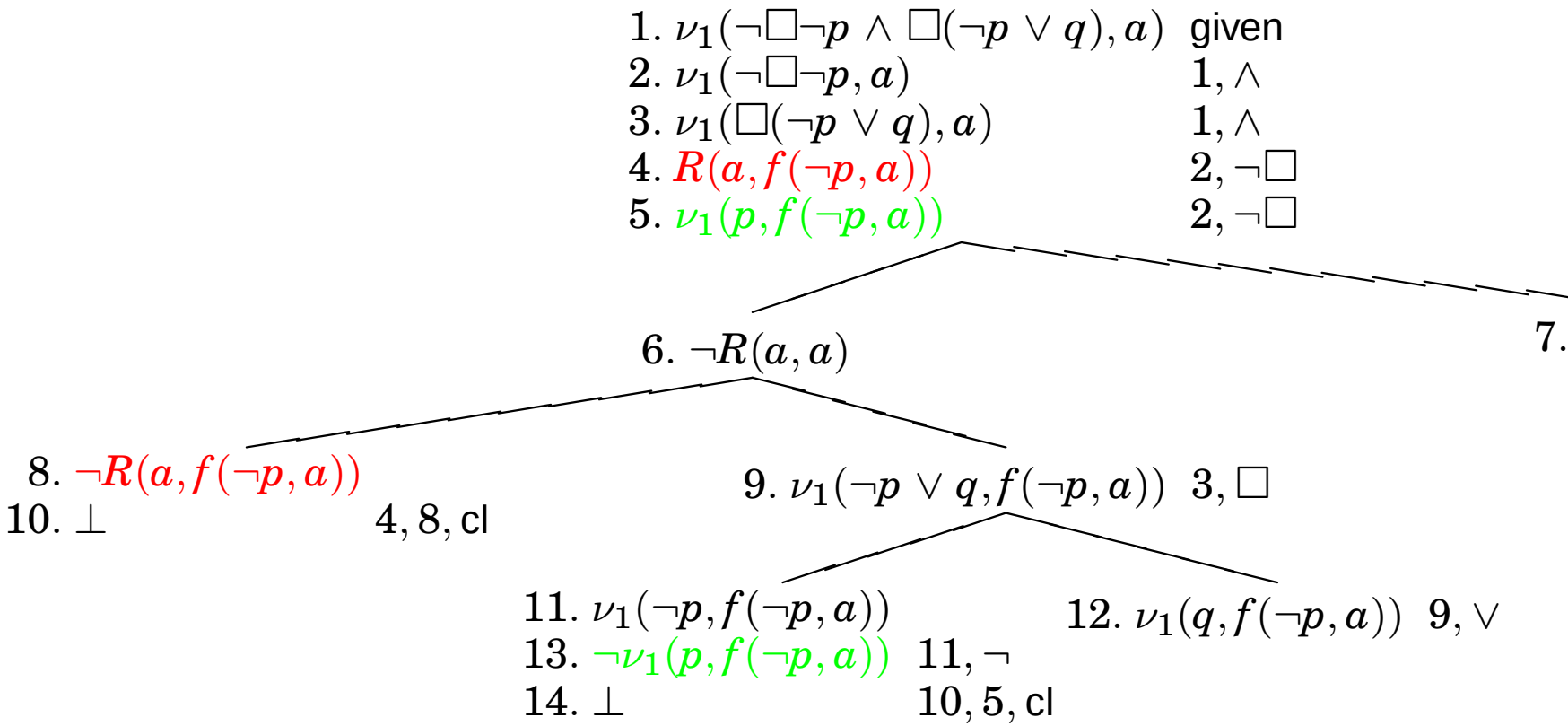
Rule application of rule $\dfrac{E_1, \ldots, E_k}{X_1 \mid \cdots \mid X_n}$:

- If the premises of a rule respectively match with some formulae $F_1, \ldots, F_k$ on a branch $\mathcal{B}$, that is, there is a substitution θ s.t. $E_1\theta = F_1, \ldots, E_k\theta = F_k$ and θ is grounding wrt. $\mathcal{B}$ for $X_1, \ldots, X_n$, then
  the rule is *applicable* to $\mathcal{B}$
- When the rule is applied, $\mathcal{B}$ is extended at the leaf with n nodes each labelled with $X_1\theta, \ldots, X_n\theta$ respectively

- θ is *grounding wrt. $\mathcal{B}$ for $X_1, \ldots, X_n$* means the inferred formulae in $X_1\theta, \ldots, X_n\theta$ are all ground (contain no variables)
- Assumption: Any rule is applied only *once* to the same set of premises and the same grounding substitution θ on each branch

A sample tableau derivation for K

- Example from before: $\Diamond p \wedge \Box(p \rightarrow q)$ is K-satisfiable
- Use the rules for S4$(@)$ minus rules for $@$, reflexivity and transitivity



Subexpression property

- A rule has the *subexpression property*, if the $\mathcal{L}$-expressions in each inferred formula are subexpressions of some $\mathcal{L}$-expression in the premises
- A tableau calculus has the *subexpression property* if each of its rules has the subexpression property

Theorem 14

The calculus generated for S4$(@)$ *has the subexpression property*

- In general:

Theorem 15

If S *is a well-defined semantic specification then* Tab_S *has the subexpression property.*

Soundness of the generated calculi

Theorem 16 (Soundness)

If S is a normalised semantic specification, then the generated calculus Tab_S is sound for the logic defined by S

- In the simplest case, S is *normalised*, if:
  - S = disjoint union of the connective definitions S^0 and the background theory S^b
  - The only $\mathcal{L}$-expressions occuring in S^b are atomic sort n expressions ($n \geq 1$)
- Recall, it suffices to prove: Every rule $X / X_1 \mid \ldots \mid X_n$ is sound, i.e.:

$$X \text{ is satisfiable} \implies \text{one of the } X_i \text{ is satisfiable}$$

- This follows easily (why?)

Completeness of the generated calculi

Theorem 17 (Refutational completeness)

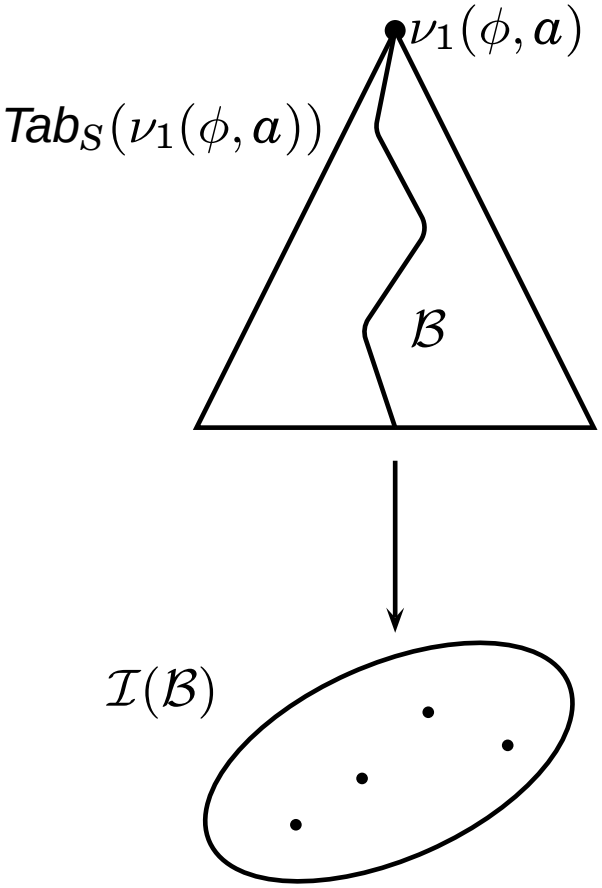
If S is a well-defined semantic specification then the generated calculus Tab_S is (constructively) complete.

- In the simplest case, a semantic specification S is *well-defined*, if:
  - S is normalised
  - The ordering $\prec$ on $\mathcal{L}$-formulae induced by S^0 is well-founded

    $\phi \prec \psi$ iff ϕ occurs on the *right* and ψ on the *left* in a connective definition equivalence

Idea of the completeness proof

Show that from any fully expanded, open branch $\mathcal{B}$ it is possible to 'constructively extract' a model $\mathcal{I}(\mathcal{B})$ of $\nu_1(\phi, a)$ that is an S-model



- Domain of $\mathcal{I}(\mathcal{B})$:

  $\Delta^{\mathcal{I}(\mathcal{B})}$ = set of ground terms occurring on $\mathcal{B}$

- Interpretation of atomic formulae with p a constant:

$$\nu_n^{\mathcal{I}(\mathcal{B})}(p, \bar{t}) = \text{true} \iff \nu_n(p, \bar{t}) \in \mathcal{B}$$
$$P^{\mathcal{I}(\mathcal{B})}(p, \bar{t}) = \text{true} \iff P(p, \bar{t}) \in \mathcal{B}$$

- Lift by induction over $\prec$ to arbitrary $FO(\mathcal{L})$-formulae and show that $\mathcal{I}(\mathcal{B})$ is an S-model of $\nu_1(\phi, a)$

By the main theorem for $\mathcal{L}$-open formulae there is a corresponding

Soundness and completeness of the calculus generated for S4(@)

- Decomposition rules

$$\frac{\nu_1(\bot,x)}{\bot} \qquad \frac{\neg\nu_1(\bot,x)}{\neg\bot} \qquad \frac{\nu_1(\neg p,x)}{\neg\nu_1(p,x)} \qquad \frac{\neg\nu_1(\neg p,x)}{\nu_1(p,x)}$$

$$\frac{\nu_1(p\vee q,x)}{\nu_1(p,x)\mid \nu_1(q,x)} \qquad \frac{\neg\nu_1(p\vee q,x)}{\neg\nu_1(p,x),\ \neg\nu_1(q,x)}$$

$$\frac{\nu_1(\Box p,x)}{\neg R(x,y)\mid \nu_1(p,y)} \qquad \frac{\neg\nu_1(\Box p,x)}{R(x,f(p,x)),\ \nu_1(p,f(p,x))}$$

$$\frac{\nu_1(@_i p,x)}{\nu_1(p,\nu_0(i))} \qquad \frac{\neg\nu_1(@_i p,x)}{\neg\nu_1(p,\nu_0(i))}$$

- Theory rules $\qquad \dfrac{}{R(x,x)} \qquad \dfrac{}{\neg R(x,y)\mid \neg R(y,z)\mid R(x,z)}$

- Closure rules $\qquad \dfrac{\nu_1(p,x),\ \neg\nu_1(p,x)}{\bot} \qquad \dfrac{R(x,y),\ \neg R(x,y)}{\bot}$

Corollary

The calculus (with or without the default equality rules) is sound and (constructively) complete for S4(@)

Refining tableau rules

3 Day 3: Tableau calculus synthesis & rule refinement

- From semantic specification to tableau calculus
- Soundness and completeness of the generated calculi
- Refining tableau rules
- Decreasing branching by moving conclusions to premises
- Standard proofs of completeness and conditions for rule refinements
- Expressing domain sort in the language of logic

Generally the degree of branching of the generated rules is higher than is necessary. Furthermore, representation of the generated rules involves the additional symbols of the language $FO(\mathcal{L})$ creating a syntactic overhead which may not always be justified. To address these problems we introduce two techniques for refining the generated rules.

Two Refinements

- Decreasing branching by moving rule conclusions to premises.
- Using logic expressiveness to eliminate symbols of domain sort.

Decreasing Branching Factor by Moving Conclusions to Premises

- Let given tableau calculus *Tab* contains the rule:

$$r = \frac{X_0}{X_1 \mid \cdots \mid X_m}.$$

- Let $X_1 = \{\psi_1, \ldots, \psi_k\}$. Define:

$$r_j \stackrel{\text{def}}{=} \frac{X_0 \cup \{\sim\psi_j\}}{X_2 \mid \cdots \mid X_m} (j = 1, \ldots, k).$$

where

$$\sim\phi \stackrel{\text{def}}{=} \begin{cases} \psi, & \phi = \neg\psi, \\ \neg\phi, & \text{otherwise.} \end{cases}$$

- Refined tableau calculus:

$$\mathit{Tab}' \stackrel{\text{def}}{=} (\mathit{Tab} \setminus \{r\}) \cup \{r_1, \ldots, r_k\}.$$

Soundness and Constructive Completeness of Refined Tableau Calculus

- r_j are derivable from r.
- If r is sound then r_j are sound.
- In general, r is not derivable in *Tab*$'$.
- Sufficient condition for admissibility of r in *Tab*$'$:

For every open branch $\mathcal{B}$ in a *Tab*$'$-tableau, if $E_1, \ldots, E_l$ are reflected in $\mathcal{I}(\mathcal{B})$ then

(†) $X_0(\overline{E}, \overline{t}) \subseteq \mathcal{B}$ implies $\mathcal{I}(\mathcal{B}) \models X_i(\overline{E}, \overline{t})$ for some $i = 1, \ldots, m$.

Theorem 18 (Refinement)

1. *Tab*$'$ *is sound whenever Tab is sound.*
2. *If Tab is constructively complete and the condition* (†) *holds for any open branch* $\mathcal{B}$ *in any Tab*$'$*-tableau then Tab*$'$ *is constructively complete.*

Simple Example: Transitivity Rule

- The transitivity rule

$$\frac{}{\neg R(x,y) \mid \neg R(y,z) \mid R(x,z)}$$

can be converted into the rule

$$\frac{R(x,y),\ R(y,z)}{R(x,z)}.$$

- The condition (†) takes the following form.

  If terms t_0, t_1, t_2 occur in $\mathcal{B}$

  then one of $\neg R(t_0,t_1), \neg R(t_1,t_2), R(t_0,t_2)$

  is true in $\mathcal{I}(\mathcal{B})$.

- If $R(t_0,t_1)$ is not in $\mathcal{B}$ then $\mathcal{I}(\mathcal{B}) \models \neg R(t_0,t_1)$ by reflection of atomic expressions.
- Similarly if $R(t_1,t_2)$ is not in $\mathcal{B}$.
- If both $R(t_0,t_1)$ and $R(t_1,t_2)$ are in $\mathcal{B}$ then, by the refined rule, $R(t_0,t_2)$ is in $\mathcal{B}$ and, hence, $\mathcal{I}(\mathcal{B}) \models R(t_0,t_2)$.

Negative Example

- In absence of other rules which can be applied to disjunctive formulae, the standard rule for disjunction

$$\frac{\nu_1(p \vee q, x)}{\nu_1(p, x) \mid \nu_1(q, x)}$$

cannot be replaced by the rule

$$\frac{\nu_1(p \vee q, x), \quad \neg\nu_1(p, x)}{\nu_1(q, x)}$$

- Given a formula $\nu_1(p \vee q, a)$, the branch $\mathcal{B}$ containing only $\nu_1(p \vee q, a)$ is fully expanded.
- The model $\mathcal{I}(\mathcal{B})$ reflects the expressions p and q.
- $\nu_1(p \vee q, a) \in \mathcal{B}$
- However $\mathcal{I}(\mathcal{B}) \not\models \nu_1(p, a)$ and $\mathcal{I}(\mathcal{B}) \not\models \nu_1(q, a)$.
- The condition (†) fails for $\mathcal{B}$ and p and q!

A Stronger Refinement Condition

- Suppose *Tab* has subexpression property.
- Then the condition (†) is (inductively) implied by the following condition:

(‡) if $X_0(\overline{E},\overline{t}) \subseteq \mathcal{B}$ and $\mathcal{I}(\mathcal{B}) \not\models X_1(\overline{E},\overline{t})$ then $X_i(\overline{E},\overline{t}) \subseteq \mathcal{B}$, for some $i = 2,\ldots,m$.

Example: $\Box$-Rule

- The rule

$$\frac{\nu_1(\Box p, x)}{\neg R(x,y) \mid \nu_1(p,y)}$$

can be replaced with the rule:

$$\frac{\nu_1(\Box p, x),\ \ R(x,y)}{\nu_1(p,y)}.$$

- The condition ($\ddagger$) takes the following form.

If $\nu_1(\Box\phi, t) \in \mathcal{B}$ and $\mathcal{I}(\mathcal{B}) \models R(t,t')$ then $\nu_1(\phi,t') \in \mathcal{B}$.

Standard Proof of Completeness with Rule Refinements

- Standard Completeness Theorem for S4$(@)$ proved by induction on the ordering $\prec$:

Lemma 19

1. *If* $@_i\phi$ *is in* $\mathcal{B}$ *then* $\mathcal{I}(\mathcal{B}), i \models \phi$ *for any formula* ϕ.
2. *For every formula* ϕ, *if* $(i,j) \in R^{\mathcal{I}(\mathcal{B})}$ *and* $@_i\Box\phi \in \mathcal{B}$ *then* $@_j\phi$ *appears in* $\mathcal{B}$.

- Compare with the condition ($\ddagger$) for $\Box$-rule:

$$\text{If } \nu_1(\Box\phi, t) \in \mathcal{B} \text{ and } \mathcal{I}(\mathcal{B}) \models R(t, t') \text{ then } \nu_1(\phi, t') \in \mathcal{B}.$$

- Standard proof of completeness includes proof of admissibility of “refined” rules.

Example: Refining Tableau Calculus for S4(@)

Decomposition rules

$$\frac{\nu_1(\neg p, x)}{\neg\nu_1(p, x)} \qquad \frac{\neg\nu_1(\neg p, x)}{\nu_1(p, x)}$$

$$\frac{\nu_1(p \vee q, x)}{\nu_1(p, x) \mid \nu_1(q, x)} \qquad \frac{\neg\nu_1(p \vee q, x)}{\neg\nu_1(p, x),\ \neg\nu_1(q, x)}$$

$$\frac{\nu_1(\Box p, x)}{\neg R(x, y) \mid \nu_1(p, y)} \rightsquigarrow \frac{\nu_1(\Box p, x),\ R(x, y)}{\nu_1(p, y)}$$

$$\frac{\neg\nu_1(\Box p, x)}{R(x, f(p, x)),\ \neg\nu_1(p, f(p, x))}$$

$$\frac{\nu_1(@_i p, x)}{\nu_1(p, \nu_0(i))} \qquad \frac{\neg\nu_1(@_i p, x)}{\neg\nu_1(p, \nu_0(i))}$$

Theory rules

$$\frac{}{R(x, x)} \qquad \frac{}{\neg R(x, y) \mid \neg R(y, z) \mid R(x, z)} \rightsquigarrow \frac{R(x, y),\ R(y, z)}{R(x, z)}$$

Closure rules

$$\frac{\nu_1(p, x),\ \neg\nu_1(p, x)}{\bot} \qquad \frac{\nu_1(\bot, x)}{\bot}$$

+ refined tableau rules for the equality axioms

Summary

- The refinement reduces the branching factor of the calculus.
- Thus, it increases performance of tableau algorithms based on the tableau calculus.
- The refinement is implicit in the standard tableau approach.
- The standard completeness proof verifies the admissibility of implicit refinements.
- The refinement works only with respect to the whole calculus!

Encoding Domain Sort

- Let L be a logic.
- Suppose there are expressions $A_n^+(p, i_1, \ldots, i_n)$, $A_n^-(p, i_1, \ldots, i_n)$, $B_P^+(i_1, \ldots, i_n)$, $B_P^-(i_1, \ldots, i_n)$ of the primary sort of L such that

$$\forall S_L \models \forall x \left(\nu_1(A_n^+, x) \rightarrow \nu_n(p, \nu_0(i_1), \ldots, \nu_0(i_n))\right)$$
$$\forall S_L \models \forall x \left(\nu_1(A_n^-, x) \rightarrow \neg\nu_n(p, \nu_0(i_1), \ldots, \nu_0(i_n))\right)$$
$$\forall S_L \models \forall x \left(\nu_1(B_P^+, x) \rightarrow P(\nu_0(i_1), \ldots, \nu_0(i_n))\right)$$
$$\forall S_L \models \forall x \left(\nu_1(B_P^-, x) \rightarrow \neg P(\nu_0(i_1), \ldots, \nu_0(i_n))\right)$$

- In particular:

$$\forall S_L \models \forall x \left(\nu_1(B_{\approx}^+, x) \rightarrow \nu_0(i_1) \approx \nu_0(i_2)\right),$$
$$\forall S_L \models \forall x \left(\nu_1(B_{\approx}^-, x) \rightarrow \nu_0(i_1) \not\approx \nu_0(i_2)\right).$$

- It is possible to eliminate symbols and predicates of the domain sort.

Domain Symbol Elimination

The tableau calculus *Tab′′* obtained from *Tab*:

- Let $i_1, \ldots, i_n$ be fresh variables of the sort 0.
- Replace occurrences $\nu_n(E, \overline{x})$ for $A_n^+(E, \overline{i})$.
- Replace occurrences $\neg\nu_n(E, \overline{x})$ for $A_n^-(E, \overline{i})$.
- Replace occurrences $P(\overline{x})$ for $B_P^+(\overline{i})$.
- Replace occurrences $\neg P(\overline{x})$ for $B_P^-(\overline{i})$.

Theorem 20

Let Tab be a sound and complete tableau calculus for a logic L. Then Tab′′ is sound and complete.
If, in addition, Tab is constructively complete then Tab′′ is also constructively complete for L.

Dealing with (Skolem) Functions

- There may not be function symbols in the object language $\mathcal{L}$ which correspond to (Skolem) functions.
- Introducing new connectives f' into $\mathcal{L}$ for every function f (including constants) of $FO(\mathcal{L})$ so that for any $p_1, \ldots, p_m, i_1, \ldots, i_n$, the expression $f'(\overline{p}, i_1, \ldots, i_n)$ is of sort 0 and its semantics is defined by

$$\nu_0(f'(\overline{p}, i_1, \ldots, i_n)) \stackrel{\text{def}}{=} f(\overline{p}, \nu_0(i_1), \ldots, \nu_0(i_n)).$$

Example: Extending Language of S4$(@)$

- Extend the language of S4$(@)$ to S4$(@, \{\cdot\})$ (full hybrid S4):
  - $\{\cdot\}$ — "singleton" connective of sort $0 \mapsto 1$;
  - Semantical definition:

$$\forall x\ (\nu_1(\{i\}, x) \equiv \nu_0(i) \approx x).$$

- Introduce a connective f' of the sort $(1, 0) \mapsto 0$ which correspond to the Skolem function f:

$$\nu_0(f'(p, i)) \stackrel{\text{def}}{=} f(p, \nu_0(i)).$$

- Expressions which defines the predicates R, $\approx$ and ν_1:

$$A_1^+(p, i) \stackrel{\text{def}}{=} @_i p, \qquad A_1^-(p, i) \stackrel{\text{def}}{=} @_i \neg p,$$
$$B_R^+(i, j) \stackrel{\text{def}}{=} @_i \Diamond \{j\}, \qquad B_R^-(i, j) \stackrel{\text{def}}{=} @_i \neg \Diamond \{j\},$$
$$B_\approx^+(i, j) \stackrel{\text{def}}{=} @_i \{j\}, \qquad B_\approx^-(i, j) \stackrel{\text{def}}{=} @_i \neg \{j\}.$$

Example: Refined Tableau Rules for S4$(@, \{\cdot\})$

Decomposition rules

$$\frac{\nu_1(\neg p, x)}{\neg\nu_1(p, x)} \rightsquigarrow \frac{@_i\neg p}{@_i\neg p} \qquad \frac{\neg\nu_1(\neg p, x)}{\nu_1(p, x)} \rightsquigarrow \frac{@_i\neg\neg p}{@_i p}$$

$$\frac{\nu_1(p \vee q, x)}{\nu_1(p, x) \mid \nu_1(q, x)} \rightsquigarrow \frac{@_i(p \vee q)}{@_i p \mid @_i q} \qquad \frac{\neg\nu_1(p \vee q, x)}{\neg\nu_1(p, x),\ \neg\nu_1(q, x)} \rightsquigarrow \frac{@_i\neg(p \vee q)}{@_i\neg p,\ @_i\neg q}$$

$$\frac{\nu_1(\Box p, x),\ R(x, y)}{\nu_1(p, y)} \rightsquigarrow \frac{@_i\Box p,\ @_i\Diamond\{j\}}{@_j p}$$

$$\frac{\neg\nu_1(\Box p, x)}{R(x, f(p, x)),\ \neg\nu_1(p, f(p, x))} \rightsquigarrow \frac{@_i\neg\Box p}{@_i\Diamond\{f'(p, i)\},\ @_{f'(p,i)}\neg p}$$

$$\frac{\nu_1(@_i p, x)}{\nu_1(p, \nu_0(i))} \rightsquigarrow \frac{@_i @_j p}{@_j p} \qquad \frac{\neg\nu_1(@_i p, x)}{\neg\nu_1(p, \nu_0(i))} \rightsquigarrow \frac{@_i\neg @_j p}{@_j\neg p}$$

$$\frac{\nu_1(\{i\}, x)}{\nu_0(i) \approx x} \rightsquigarrow \frac{@_i\{j\}}{@_i\{j\}} \qquad \frac{\neg\nu_1(\{i\}, x)}{\nu_0(i) \not\approx x} \rightsquigarrow \frac{@_i\neg\{j\}}{@_i\neg\{j\}}$$

Theory rules

$$\frac{}{R(x, x)} \rightsquigarrow \frac{}{@_i\Diamond\{i\}} \qquad \frac{R(x, y),\ R(y, z)}{R(x, z)} \rightsquigarrow \frac{@_i\Diamond\{j\},\ @_j\Diamond\{k\}}{@_i\Diamond\{k\}}$$

+closure rules

Equality Congruence Rules for S4$(@, \{\cdot\})$

$$\frac{@_i\{j\}}{@_j\{i\}} \qquad \frac{@_i\neg\{j\}}{@_j\{j\}} \qquad \frac{@_i p}{@_i\{i\}} \qquad \frac{@_i\neg\Diamond\{j\}}{@_j\{j\}} \qquad \frac{@_i p,\ \ @_i\{j\}}{@_j p}$$

$$\frac{@_i\Diamond\{j\},\ \ @_j\{k\}}{@_i\Diamond\{k\}} \qquad \frac{@_i\{j\}}{@_{f'(p,i)}\{f'(p,j)\}}$$

Summary

- Two rule refinements were introduced.
- The first refinement allows to reduce branching factor of a tableau calculus and increase performance of tableau algorithms based on the calculus.
- It can be applied iteratively.
- The second refinement reduce an overhead of using symbols of the domain sort by expressing the domain sort within the language of the logic.
- Both refinement preserve soundness, (constructive) completeness and the subexpression property.

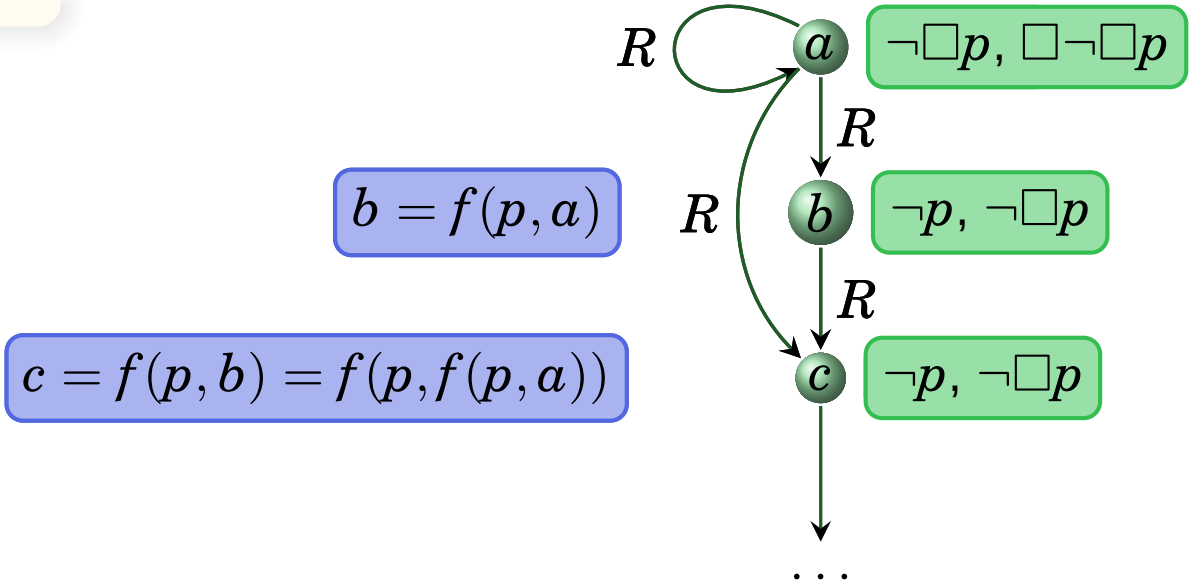
Outline

4. Day 4: Blocking mechanisms and the finite model property
   - Ensuring termination using standard loop-checking mechanisms
   - The unrestricted blocking mechanism
   - Proving termination based on the finite model property
   - Turning ground semantic tableau calculi into deterministic procedures

We consider several known loop-checking and blocking mechanisms for detecting repetitions in tableau derivations and achieving termination. We demonstrate a problem of these mechanisms for detecting repetitions in expressive description logics. We introduce an unrestricted blocking mechanism and prove that it is general enough to ensure termination for logics with the finite model property. We also show how to simulate known loop-checking mechanisms by restricting the unrestricted blocking mechanism.

Motivating Example

- Satisfiability of the formula $\Box\neg\Box p$ for S4$(@)$.



- It is required to detect repetitions in partially constructed models!

Terminating Tableau Calculus

- A tableau calculus is *(weakly) terminating* iff

  any fully expanded tableau for any satisfiable set of expressions has a *finite branch*.

- In order to ensure termination various blocking mechanisms are required.

Blocking

- Blocking/loop-checking is a mechanism for preventing expansion of nodes which can cause a repetitive derivation.
- Standard loop-checking mechanisms combine the following techniques:
  - Subset and equality blocking.
  - Ancestor and anywhere blocking.
  - Static and dynamic blocking.
- Many other techniques:
  - Pairwise blocking.
  - Pattern-base blocking.
  - Blocking via reuse of terms.
- Unrestricted blocking mechanism.

Most Common Blocking Techniques

- For every pair of nominals (or labels) i, j verify blocking conditions.
- If blocking conditions are true for i and j, block one of i, j for application of rules which can generate a new nominal term.
- Unblock a nominal if blocking conditions become false.
- If blocks on individuals are never undone then blocking is called *static*. Otherwise it is called *dynamic*.
- If blocking conditions includes the condition that j is ancestor of i, then the blocking is called *ancestor* blocking. Otherwise it is called *anywhere* blocking.
- In hybrid logic, blocking mechanisms require access to a set of expressions $\tau(i)$ associated with given nominal i:

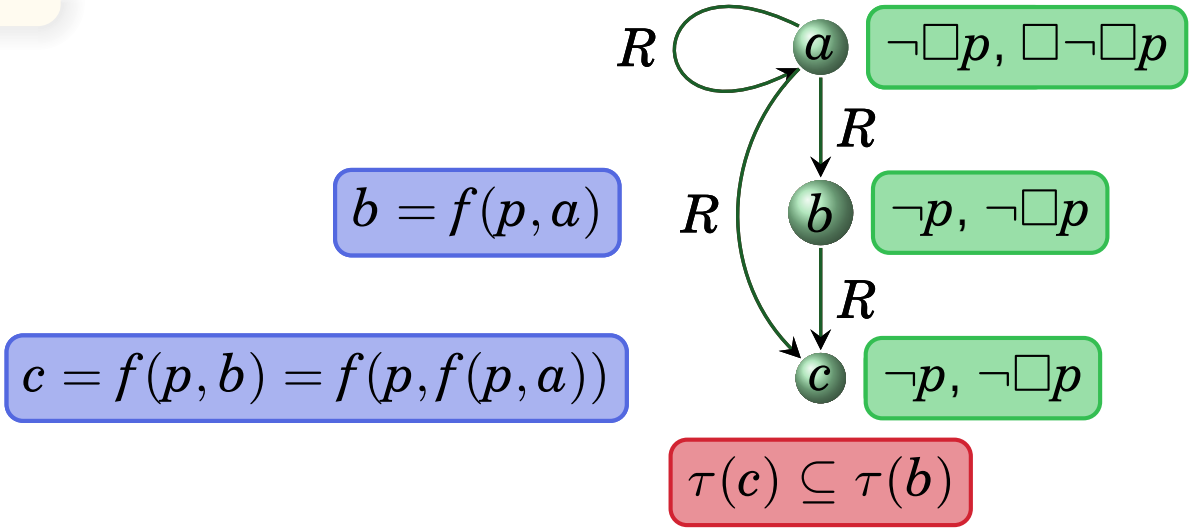
$$\tau(i) \stackrel{\text{def}}{=} \{\phi \mid @_i\phi \text{ is in } \mathcal{B}\}$$

- In the language of the framework (for a domain term t):

$$\tau(t) \stackrel{\text{def}}{=} \{\phi \mid \nu_1(\phi, t) \text{ is in } \mathcal{B}\}$$

Example: Subset Blocking

- Blocking condition: $\tau(x) \subseteq \tau(y)$.
- Example: $\Box\neg\Box p$ for S4$(@)$.



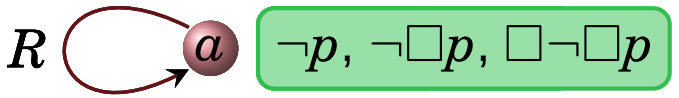
Example: Reusing Terms

- Use a special form of the $\neg\Box$-rule:

$$\frac{\nu_1(\neg\Box p, x)}{R(x,t_1),\ \ \nu_1(\neg p,t_1) \mid \cdots \mid R(x,t_n),\ \ \nu_1(\neg p,t_n) \mid R(x,f(p,x)),\ \ \nu_1(\neg p,f(p,x))}$$

$t_1,\ldots,t_n$ are all domain terms in the branch.

- Example: $\Box\neg\Box p$ for S4$(@)$.



Why Standard Blocking Mechanisms Are Not Sufficient?

- Choice of blocking mechanisms heavily depend on a concrete logic and a tableau calculus.
- Soundness and completeness of a calculus need to be reproved when adding a blocking mechanism.
- Standard blocking mechanisms are not sufficient for the Boolean Modal Logic and description logics with full role negation.
- We need a more general blocking mechanism which can guarantee termination and does not destroy soundness and completeness of calculi.

Unrestricted Blocking Mechanism: Intuition

- Suppose given formula is satisfiable in a finite model.
- Since tableau derivation constructs provisional model for the formula, attempt to make the model as smaller as possible.
- In a case if model cannot be made smaller (i.e. that leads to a contradiction) backtrack.

Unrestricted Blocking Mechanism

Unrestricted Blocking Rule:

$$\text{(ub):}\ \frac{}{x \approx y \mid x \not\approx y}$$

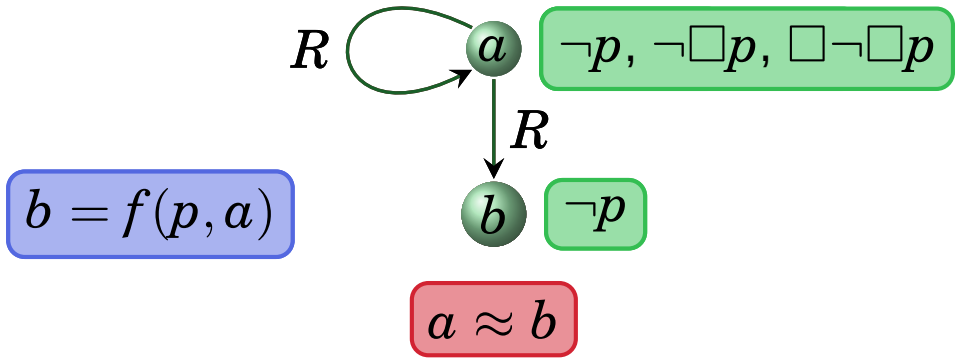
Strategy restrictions:

1. If $t \approx t'$ appears in a branch and $t < t'$[1] then all further applications of rules which introduce new terms to expressions containing t' are not performed within the branch.
2. In every open branch there is some node from which point onwards before any application of a rule introducing a new term all possible applications of the (ub) rule have been performed.

[1] $<$ reflects the order in which new terms are introduced

Example: Unrestricted Blocking

- Example: $\Box\neg\Box p$ for S4$(@)$.



Soundness and Completeness

- The unrestricted blocking rule is sound.
- Thus, it can be safely added to any complete calculus without destroying the completeness.
- Adding the unrestricted blocking rule preserves constructive completeness of the calculus.

Theorem 21

If Tab is sound and (constructively) complete tableau calculus for a logic L then Tab $+$ (ub) *is sound and (constructively) complete tableau calculus for L.*

Simulating Existing Blocking Mechanisms

- The unrestricted blocking mechanism can simulate the blocking via reusing terms.
- The converse is not true!
- It can simulate subset and equality blocking, and other blocking mechanisms:
  - Add conditions for blocking as constraints on application of the unrestricted blocking rule, e.g.

$$\frac{\tau(x) \subseteq \tau(y)}{x \approx y \mid x \not\approx y}$$

  - Use the rule application strategy which is needed for the simulated blocking mechanism. (E.g. apply all ‘local’ rules before the others).
- Work in progress.

Subexpression Operator

- sub an operator which maps sets of expressions to sets of expressions.
- A tableau calculus *Tab* is *compatible with* sub, iff for every set of expressions N, all $\mathcal{L}$-expressions occurring in a fully expanded tableau derivation *Tab*(N) belong to sub(N).
- sub is finite if sub(N) is finite for every finite N.

Ensuring Compatibility with sub

- If S is finite then the induced ordering $\prec$ is finitely branching.
- If S is well-defined then $\prec$ is well-founded.
- Thus, if S is finite and well-defined then $\text{sub}_{\preceq}$ is finite.
- Every rule respects the ordering $\prec$.
- Thus, Tab_S is compatible with $\text{sub}_{\preceq}$.

Theorem 22

Suppose $\prec$ is induced by a finite and well-defined semantic specification S. Then,
Tab_S is compatible with $\text{sub}_{\preceq}$ and $\text{sub}_{\preceq}$ is finite.

Corollary

The calculus generated for S4(@) is compatible with $\text{sub}_{\preceq}$ and $\text{sub}_{\preceq}$ is finite.

Termination Criterion

Lemma 23 (Small Branch Lemma)

Let Tab be a sound and constructively complete calculus for a logic L which includes the unrestricted blocking rule.
Let Tab(N) be a tableau for an input set N which is satisfiable in a model $\mathcal{I}$.
Then there is a branch $\mathcal{B}$ in Tab(N) such that $|\Delta^{\mathcal{I}(\mathcal{B})}| \leq |\Delta^{\mathcal{I}}|$.

Theorem 24 (Finite Open Branch Theorem)

If a set of formulae N is satisfiable in a finite model and Tab is compatible with a finite closure operator sub *then there is a finite open branch in Tab(N).*

Theorem 25 (Termination)

Let Tab be a sound and constructively complete tableau calculus for a logic L which is compatible with a finite closure operator sub *for L-expressions. Then Tab $+$ (ub) is terminating if and only if L has the finite model property.*

Corollary

Let $Tab_{S4(@)}$ be a generated calculus for S4$(@)$. Then $Tab_{S4(@)} +$ (ub) is sound, (constructively) complete and terminating tableau calculus for S4$(@)$.

Summary

- A necessity of using blocking is discussed.
- Several blocking techniques are demonstrated.
- A new generic blocking mechanism is introduced.
- It ensures termination of a calculus provided that the calculus is sound and constructively complete for a logic which has a finite model property.
- It can simulate other blocking mechanisms such that subset and equality blocking, term-reusing blocking, etc.

Non-determinism present in tableau calculi

- The application of tableau rules is non-deterministic
- At any point in the derivation process we have complete flexibility in choosing:
  - which branch to select and expand next
  - which rule to apply next
  - which formula to select

$\Longrightarrow$ Tableau calculi provide *non-deterministic* procedures

Terminating tableau calculi provide *non-deterministic* decision procedures

Problem

How to turn the generated tableau calculi into deterministic procedures?

Turning tableau calculi into deterministic procedures

Ensuring completeness:

- If N is unsatisfiable, then any tableau $\mathit{Tab}(N)$ constructed non-deterministically for it is closed
- Problem: Does this imply that every *deterministic* procedure starting from N constructs a closed tableau?

Problem associated with rules of universal extent

- Rules with universal extent (i.e., γ-rules) can cause problems, regardless as to whether we use the unrefined or refined versions
- Unrefined $\Box$-rule:

$$\frac{\nu_1(\Box p, x)}{\neg R(x,y) \mid \nu_1(p,y)}$$

Applicable to the same formula $\nu_1(\Box\phi, t)$ on a branch for each domain term t occuring on that branch
(Recall, the substitution used in a rule appication must be grounding)

- Refined $\Box$-rule:

$$\frac{\nu_1(\Box p, x),\ \ R(x,y)}{\nu_1(p,y)}$$

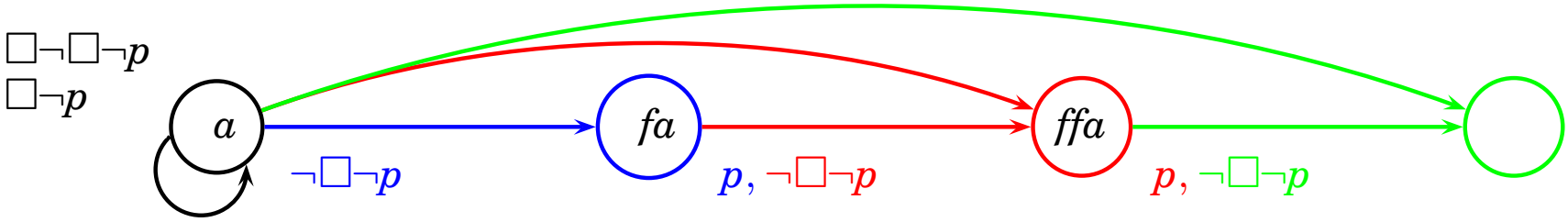
Applicable to the same formula $\nu_1(\Box\phi, t)$ on a branch for each $R(t,t')$ occuring on that branch

Problem associated with rules of universal extent (cont'd)

- Example: $\{\ \Box\Diamond p,\ \ \Box\neg p\ \}$ is S4-unsatisfiable

1.	$a : \Box\neg\Box\neg p$	given
2.	$a : \Box\neg p$	given
3.	$(a,a) : R$	reflexivity
4.	$a : \neg\Box\neg p$	$1, 3, \Box$
5.	$(a, f(\neg p, a)) : R$	$4, \neg\Box$
6.	$f(\neg p, a) : p$	$4, \neg\Box$
~~7.~~	$(a, f(\neg p, a)) : R$	$3, 5$, transitivity
8.	$f(\neg p, a) : \neg\Box\neg p$	$1, 5, \Box$
9.	$(f(\neg p, a), f(\neg p, f(\neg p, a))) : R$	$8, \neg\Box$
10.	$f(\neg p, f(\neg p, a)) : p$	$8, \neg\Box$
11.	$(a, f(\neg p, f(\neg p, a))) : R$	$5, 9$, transitivity
12.	$f(\neg p, f(\neg p, a)) : \neg\Box\neg p$	$1, 11, \Box$
	$\vdots$	
	ad infinitum	

- This is an unfair derivation; formula 2. is indefinitely ignored



Fairness

- We need a notion of fairness:

- A tableau procedure is *fair* if:
  When a rule is applicable to a formula then the rule is eventually applied to this formula on every branch on which it occurs
  (unless the branch is closed and an open, fully expanded branch has already been found)

Theorem 26

If Tab is refutationally complete then every fair (deterministic) tableau procedure based on Tab is refutationally complete.

- Possible way of guaranteeing fairness:
  Giving γ-rules and γ-formulae equal priority ensures completeness
  $\Longrightarrow$ store in a queue data-structures

Turning tableau calculi into deterministic procedures (cont'd)

Ensuring strong termination:

- If N is a finite, satisfiable set, then any tableau $\mathit{Tab}(N)$ constructed non-deterministically for it is finite
- Problem: Does this imply that every *deterministic* procedure starting from N constructs a finite open tableau?

Yes, for every fair derivation

Ensuring weak termination:

- If N is a finite, satisfiable set, then any $\mathit{Tab}(N)$ constructed non-deterministically for it has a finite open, fully expanded branch
- Problem: Does this imply that every *deterministic* procedure starting from N constructs a finite open, fully expanded branch?

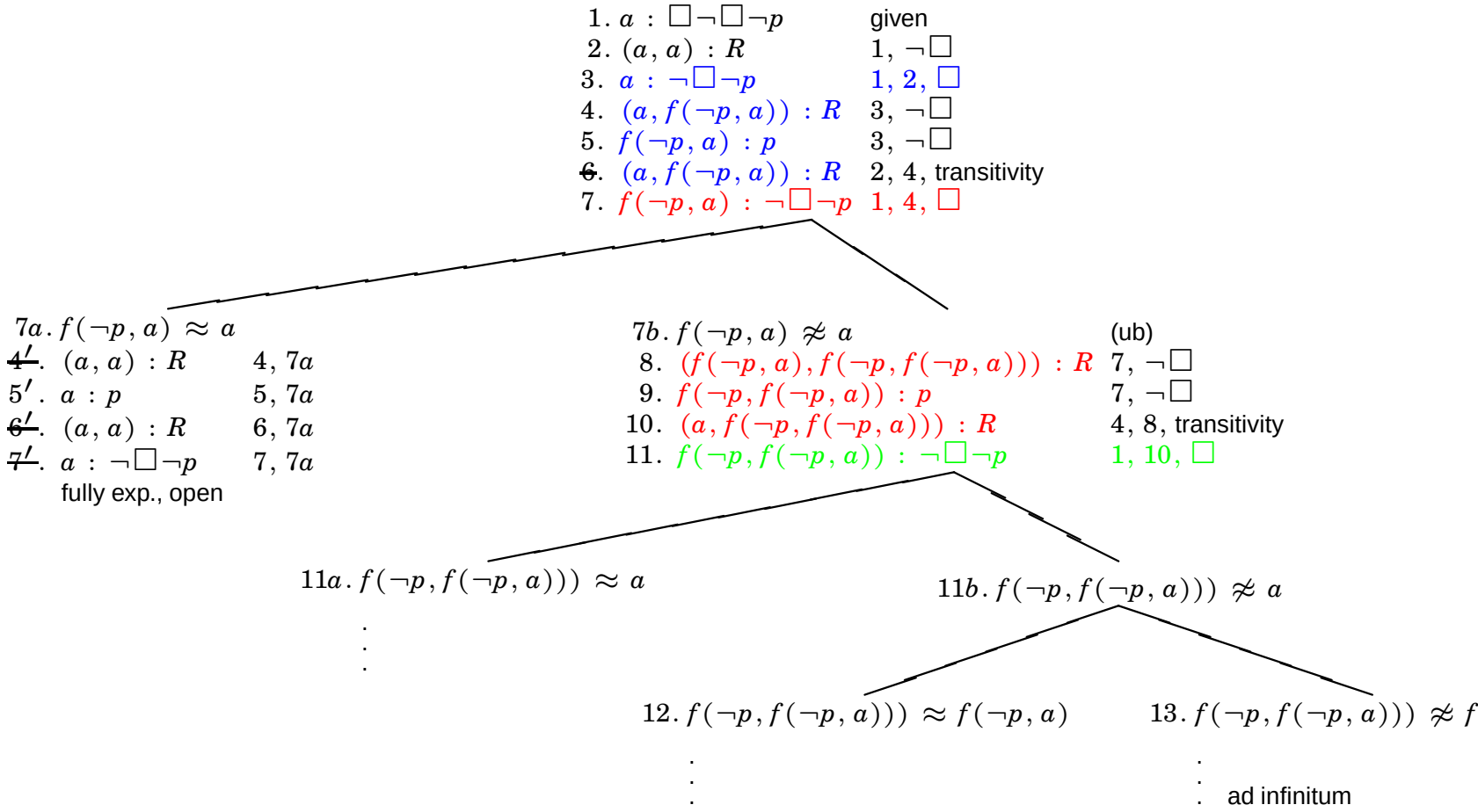
Problem associated with unrestricted blocking

$$\frac{}{x \approx y \mid x \not\approx y}$$

- Always choosing the right branch at (ub) branching points is like not using blocking at all
- Example: $\{ \Box\Diamond p \}$ is S4-satisfiable

1.	$a : \Box\neg\Box\neg p$	given
2.	$(a,a) : R$	reflexivitity
3.	$a : \neg\Box\neg p$	$1, 2, \Box$
4.	$(a, f(\neg p, a)) : R$	$3, \neg\Box$
5.	$f(\neg p, a) : p$	$3, \neg\Box$
~~6.~~	$(a, f(\neg p, a)) : R$	$2, 4$, transitivity
7.	$f(\neg p, a) : \neg\Box\neg p$	$1, 4, \Box$
8.	$(f(\neg p, a), f(\neg p, f(\neg p, a))) : R$	$7, \neg\Box$
9.	$f(\neg p, f(\neg p, a)) : p$	$7, \neg\Box$
10.	$(a, f(\neg p, f(\neg p, a))) : R$	$4, 8$, transitivity
11.	$f(\neg p, f(\neg p, a)) : \neg\Box\neg p$	$1, 10, \Box$
	$\vdots$	ad infinitum

Problem associated with unrestricted blocking (cont'd)



- Constructed by using depth-first right-to-left search strategy
- Always selecting the right-most branch is unfair; the right-most branch is infinite

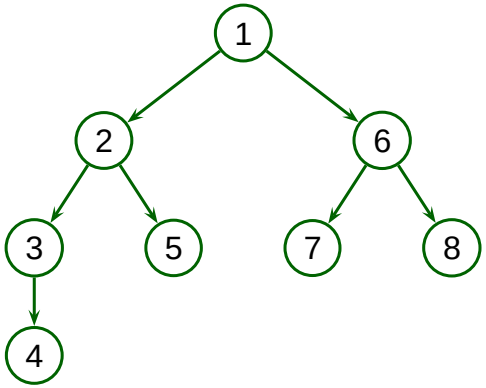
Problem where we only have weak termination

- There is an example (due to Hilverd Reker) for a decidable logic with the finite model property for which always choosing left-most branches causes non-termination
- This means we have to treat branches fairly

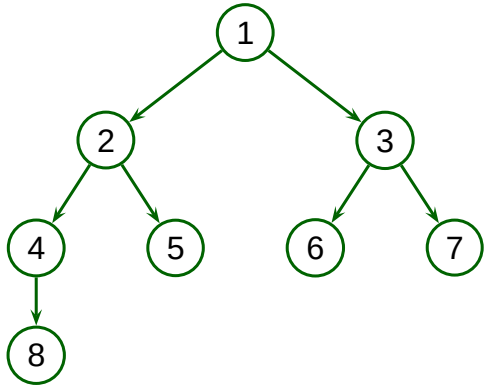
Possible ways of getting decision procedures

- For calculi for which strong termination can be shown:
  Any fair tableau procedure provides a decision procedure, including one based on *depth-first search* and a *left-to-right branch selection* strategy

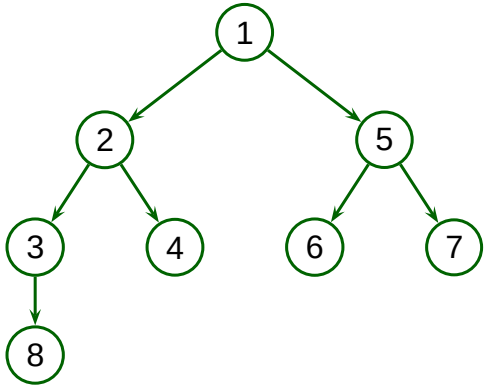
- For calculi for which only weak termination can be shown:
  Any fair tableau procedure based on *breadth-first search* strategy provides a decision procedure
  Alternatively use a *depth-first iterative deeping* or *depth-first up to maximal depth search* strategy



DFLR



BF



DFID $n = 2$

Outline

5. Day 5: More examples & METTEL
   - The modal logic of some, all and only
   - Determining the admissibility of S4-rules
   - The METTEL prover
   - Concluding remarks

The Framework: How it all works

1. Formalise the semantics of the logic
2. Verify the well-definedness conditions
3. Use the method to automatically generate a tableau calculus
   - Results ensure the calculus is sound and constructively complete
4. Refine and optimise the generated tableau calculus
5. Prove the finite model property and compatability with a finite closure operator sub for the logic, then add the unrestricted blocking mechanism to the calculus
   - Results ensure the calculus is terminating

Applications of the framework

Concrete case studies

Intuitionistic propositional logic	[TABLEAUX09, LMCS11]	
Description logic $\mathcal{SO}$	[LMCS11]	
Modal logic S4(u,+)	[ENTCS10]	today
Hybrid logic S4$(@)$		this week
Hybrid logic S4$(@, \{\cdot\})$		yesterday
K$_{(m)}(\neg)$		today
Other description logics		ongoing
Functional semantics of extensions of modal logic K		ongoing

Designed to cover

Description logics $\mathcal{ALBO}$, $\mathcal{ALBO}^{\text{id}}$
Dynamic modal logics, Peirce logic
Relational logics
. . .

The modal logic of some, all and only

- $\mathsf{K}_{(m)}(\neg)$ = the modal logic of 'some', 'all' and 'only'

- Modal formulae: $\phi, \psi \longrightarrow p \mid \bot \mid \neg\phi \mid \phi \vee \psi \mid [\alpha]\phi$
  Actions: $\alpha \longrightarrow r \mid \neg\alpha$

- Semantics: Kripke model $\mathcal{M} = (W, R, v)$

valuation mapping
R : actions $\longrightarrow 2^{W \times W}$

$\mathcal{M}, x \models p$	iff	$x \in v(p)$
$\mathcal{M}, x \not\models \bot$		
$\mathcal{M}, x \models \neg\phi$	iff	$\mathcal{M}, x \not\models \phi$
$\mathcal{M}, x \models \phi \vee \psi$	iff	$\mathcal{M}, x \models \phi$ or $\mathcal{M}, x \models \psi$
$\mathcal{M}, x \models [\alpha]\phi$	iff	for all R_α-successors y of x $\mathcal{M}, y \models \phi$
$(x, y) \in R_{\neg\alpha}$	iff	$(x, y) \notin R_\alpha$

Notation: R_α instead of $R(\alpha)$

Why is $\mathsf{K}_{(m)}(\neg)$ interesting?

- Not widely studied; no tableau calculi known
- In $\mathsf{K}_{(m)}(\neg)$ there are 3 quantifier operators:

Necessity operator: $\mathcal{M},x \models [\alpha]\phi$ iff $\forall y \left(\boldsymbol{R}_\alpha(x,y) \rightarrow \mathcal{M},y \models \phi \right)$

Possibility operator: $\mathcal{M},x \models \langle\alpha\rangle\phi$ iff $\exists y \left(\boldsymbol{R}_\alpha(x,y) \wedge \mathcal{M},y \models \phi \right)$

Sufficiency operator: $\mathcal{M},x \models \neg\langle\neg\alpha\rangle\phi$ iff $\forall y \left(\mathcal{M},y \models \phi \rightarrow \boldsymbol{R}_\alpha(x,y) \right)$

$\neg\langle\neg\alpha\rangle$ is a dual of $\langle\alpha\rangle$, like $[\alpha]$ is a dual of $\langle\alpha\rangle$; recall $[\alpha] \equiv \neg\langle\alpha\rangle\neg$

Notation	Reading
$[\alpha]\phi$	after performing α, ϕ is necessarily true
$\langle\alpha\rangle\phi$	after performing α, ϕ is possibly true
$\neg\langle\neg\alpha\rangle\phi$	ϕ is true is a sufficient condition for α to be performable
$[\alpha]\phi$	ϕ is true at all α-successors
$\langle\alpha\rangle\phi$	ϕ is true at some α-successor
$\neg\langle\neg\alpha\rangle\phi$	ϕ is true at only α-successors

- $\mathsf{K}_{(m)}(\neg)$ has the finite model property but does not have the tree model property
- $\mathsf{K}_{(m)}(\neg)$ is can be decided by translation to first-order logic and resolution

Applying the framework to $\mathsf{K}_{(m)}(\neg)$

Object language for $\mathcal{L} = \mathsf{K}_{(m)}(\neg)$

- Expressions:

  modal formulae $\quad \phi, \psi \longrightarrow p \mid \bot \mid \neg\phi \mid \phi \vee \psi \mid [\alpha]\phi$

  actions $\quad \alpha \longrightarrow r \mid \neg\alpha$

- Sorts:
  - 0 none
  - 1 for modal formulae
  - 2 for actions

Semantic specification language $FO(\mathcal{L})$

- Sorts: 1, 2 plus 3 for the domain sort
- Sorts 1 and 2 defined as above
- Sort 3: $t, t' \longrightarrow a \mid x \mid f(\alpha, \phi, t)$
- Designated symbols: $\nu_1(\phi : 1, t : 3)$
  $\nu_2(\alpha : 2, t : 3, t : 3)$

Semantic specification of $\mathsf{K}_{(m)}(\neg)$

- S^0: Connective definitions

$$\forall x\ (\nu_1(\bot, x) \equiv \bot)$$
$$\forall x\ (\nu_1(\neg p, x) \equiv \neg\nu_1(p, x))$$
$$\forall x\ (\nu_1(p \vee q), x) \equiv \nu_1(p, x) \vee \nu_1(q, x))$$
$$\forall x\ (\nu_1([r]p, x) \equiv \forall y\ (\nu_2(r, x, y) \rightarrow \nu_1(p, y)))$$
$$\forall x \forall y\ (\nu_2(\neg r, x, y) \equiv \neg\nu_2(r, x, y))$$

- $S^b = \varnothing$

Generated rules for $\mathsf{K}_{(m)}(\neg)$

- Decomposition rules:

$$\frac{\nu_1(\neg p, x)}{\neg\nu_1(p, x)} \qquad \frac{\neg\nu_1(\neg p, x)}{\nu_1(p, x)}$$

$$\frac{\nu_1(p \vee q), x)}{\nu_1(p, x) \mid \nu_1(q, x)} \qquad \frac{\neg\nu_1(p \vee q), x)}{\neg\nu_1(p, x), \ \neg\nu_1(q, x)}$$

$$\frac{\nu_1([r]p, x)}{\neg\nu_2(r, x, y) \mid \nu_1(p, y)} \qquad \frac{\neg\nu_1([r]p, x)}{\nu_2(r, x, f(r, p, x)), \ \neg\nu_1(p, f(r, p, x))}$$

$$\frac{\nu_2(\neg r, x, y)}{\neg\nu_2(r, x, y)} \qquad \frac{\neg\nu_2(\neg r, x, y)}{\nu_2(r, x, y)}$$

- Closure rules:

$$\frac{\nu_1(p, x), \ \neg\nu_1(p, x)}{\bot} \qquad \frac{\nu_2(r, x, y), \ \neg\nu_2(r, x, y)}{\bot}$$

Terminating calculus for $\mathsf{K}_{(m)}(\neg)$

- The specification is normalised and well-defined

Theorem 27

- *$Tab_{\mathsf{K}_{(m)}(\neg)}$ is sound and complete*
- *Adding the default equality rules plus unrestricted blocking gives a terminating calculus for $\mathsf{K}_{(m)}(\neg)$*

Theorem 28

- *Any fair tableau procedure based on $Tab_{\mathsf{K}_{(m)}(\neg)} + (\mathsf{eq}) + (\mathsf{ub})$ using either a BF, DFID or DFMD search strategy is a decision procedure for $\mathsf{K}_{(m)}(\neg)$*

Problem of refining the rules

- The language of $\mathsf{K}_{(m)}(\neg)$ is not expressive enough to generate rules operating on labelled formulae, but adding the $@$ and $\{\cdot\}$ operators will allow us to eliminate the ν-predicates

- Question: Can the $[\cdot]$ rule be replaced by the refined non-branching version?

$$\frac{\nu_1([r]p, x)}{\neg\nu_2(r, x, y) \mid \nu_1(p, y)} \quad \rightsquigarrow \quad \frac{\nu_1([r]p, x), \quad \nu_2(r, x, y)}{\nu_1(p, y)} \quad ?$$

- No, conditions (†) and (‡) don't hold

- Nevertheless there is a way to use the framework to refine the tableau calculus to use a non-branching $[\cdot]$ rule

An alternative semantic specification of $\mathsf{K}_{(m)}(\neg)$

- S^0: Original connective definitions

$$\forall x\ (\nu_1(\bot, x) \equiv \bot)$$
$$\forall x\ (\nu_1(\neg p, x) \equiv \neg\nu_1(p, x))$$
$$\forall x\ (\nu_1(p \vee q), x) \equiv \nu_1(p, x) \vee \nu_1(q, x))$$
$$\forall x\ (\nu_1([r]p, x) \equiv \forall y\ (\nu_2(r, x, y) \rightarrow \nu_1(p, y)))$$
$$\forall x \forall y\ (\nu_2(\neg r, x, y) \equiv \neg\nu_2(r, x, y))$$

- Alternative specification S:

$$\forall x\ (\nu_1(\bot, x) \equiv \bot)$$
$$\forall x\ (\nu_1(\neg p, x) \equiv \neg\nu_1(p, x))$$
$$\forall x\ (\nu_1(p \vee q), x) \equiv \nu_1(p, x) \vee \nu_1(q, x))$$
$$\forall x\ (\nu_1([r]p, x) \equiv \forall y\ (\nu_2(r, x, y) \rightarrow \nu_1(p, y)))$$
$$\forall x \forall y\ (\nu_2(\neg r, x, y) \equiv \neg\nu_2(r, x, y))$$
$$\forall x\ (\nu_1([\neg r]p, x) \rightarrow \forall y\ (\neg\nu_2(r, x, y) \rightarrow \nu_1(p, y)))$$

Is S well-defined?

- $\forall S^0, \forall S^b \models \forall S$ Yes
- $\prec$ induced by S is well-founded Yes
- $\forall S^0, S^b \restriction \mathsf{sub}_\prec(E) \models \forall \overline{x}\Big(\Big(\bigwedge \Phi^E_+ \rightarrow \phi^\sigma(E_1, \ldots, E_m, \overline{x})\Big) \wedge \Big(\phi^\sigma(E_1, \ldots, E_m, \overline{x}) \rightarrow \bigvee \Phi^E_-\Big)\Big)$ Yes
- S must define the semantics of the connectives, i.e. $\forall S \models \forall S^0$ Yes

Rules generated from alternative specification S

- Decomposition rules:

$$\frac{\nu_1(\neg p, x)}{\neg\nu_1(p, x)} \qquad \frac{\neg\nu_1(\neg p, x)}{\nu_1(p, x)}$$

$$\frac{\nu_1(p \vee q), x)}{\nu_1(p, x) \mid \nu_1(q, x)} \qquad \frac{\neg\nu_1(p \vee q), x)}{\neg\nu_1(p, x),\ \neg\nu_1(q, x)}$$

$$\frac{\nu_1([r]p, x)}{\neg\nu_2(r, x, y) \mid \nu_1(p, y)} \qquad \frac{\neg\nu_1([r]p, x)}{\nu_2(r, x, f(r, p, x)),\ \neg\nu_1(p, f(r, p, x))}$$

$$\frac{\nu_2(\neg r, x, y)}{\neg\nu_2(r, y, x)} \qquad \frac{\neg\nu_2(\neg r, x, y)}{\nu_2(r, y, x)}$$

$$\frac{\nu_1([\neg r]p, x)}{\nu_2(r, x, y) \mid \nu_1(p, y)}$$

- Closure rules:

$$\frac{\nu_1(p, x),\ \neg\nu_1(p, x)}{\bot} \qquad \frac{\nu_2(r, x, y),\ \neg\nu_2(r, x, y)}{\bot}$$

- Now, the condition (†) holds.

Refined alternative calculus

- Decomposition rules:

$$\frac{\nu_1(\neg p, x)}{\neg\nu_1(p, x)} \qquad \frac{\neg\nu_1(\neg p, x)}{\nu_1(p, x)}$$

$$\frac{\nu_1(p \vee q), x)}{\nu_1(p, x) \mid \nu_1(q, x)} \qquad \frac{\neg\nu_1(p \vee q), x)}{\neg\nu_1(p, x),\ \neg\nu_1(q, x)}$$

$$\frac{\nu_1([r]p, x),\ \nu_2(r, x, y)}{\nu_1(p, y)} \qquad \frac{\neg\nu_1([r]p, x)}{\nu_2(r, x, f(r, p, x)),\ \neg\nu_1(p, f(r, p, x))}$$

$$\frac{\nu_2(\neg r, x, y)}{\neg\nu_2(r, y, x)} \qquad \frac{\neg\nu_2(\neg r, x, y)}{\nu_2(r, y, x)}$$

$$\frac{\nu_1([\neg r]p, x)}{\nu_2(r, x, y) \mid \nu_1(p, y)}$$

- Closure rules:

$$\frac{\nu_1(p, x),\ \neg\nu_1(p, x)}{\bot} \qquad \frac{\nu_2(r, x, y),\ \neg\nu_2(r, x, y)}{\bot}$$

Theorem 29

Tab_S is sound and complete for $\mathsf{K}_{(m)}(\neg)$
$Tab_S + (\text{eq}) + (\text{ub})$ is a sound, complete and terminating calculus for $\mathsf{K}_{(m)}(\neg)$

Motivation

- A new logic.
- No tableau calculus known.
- Non-trivial semantics which does not immediately fit into the framework.
- A good test case for the framework.

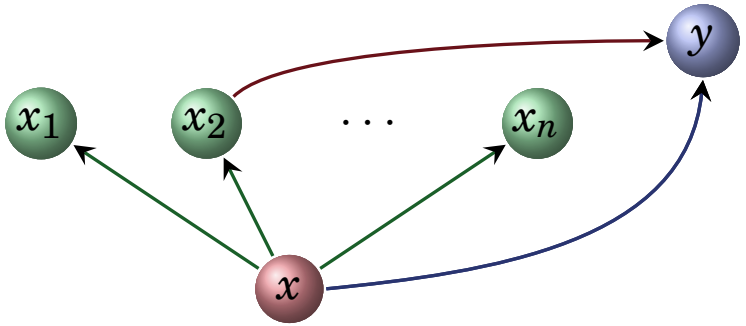
Admissible Rules

- α/β is *derivable* in L iff $\alpha \vdash_L \beta$.
- A rule is admissible for a logic if the set of theorems of the logic is closed under the rule. (Lorenzen 1955)
- α/β is *admissible* for L if for every substitution σ, $\sigma(\alpha) \in L \Longrightarrow \sigma(\beta) \in L$.
- Derivable $\Rightarrow$ admissible.
- Admissible $\not\Rightarrow$ derivable. (Harrop 1960, Mints 1976)
- Algorithm for checking rule admissibility for S4 (Rybakov 1984)

Original Algorithm

- Reduce given rule to a normal form.
- Disprove the rule α/β in a special S4-model $\mathcal{M} = (W, R, \nu)$ with the co-cover property.
- Co-cover property ($n = 0, 2, \ldots$):

$$\forall x_1 \cdots \forall x_n \exists x \left[R(x, x_1) \wedge \cdots \wedge R(x, x_n) \wedge \forall y \left(R(x, y) \rightarrow (x \approx y \vee R(x_1, y) \vee \cdots \vee R(x_n, y)) \right) \right]$$



- Show that $\mathcal{M}, x_0 \not\models \beta$ for some $x_0 \in W$ and $\mathcal{M}, x \models \alpha$ for all $x \in W$.
- 2ExpTime-algorithm!

Sequence of Steps

- Reduce the rule admissibility problem in S4 to a satisfiability problem in some special logic $S4^{u,+}$.
- Prove that $S4^{u,+}$ has the finite model property.
- Prove that $S4^{u,+}$ has a well-defined semantic specification.
- Apply the framework: generate and refine a tableau calculus for $S4^{u,+}$.
- Add the unrestricted blocking mechanism to ensure termination of the calculus.
- Combine the reduction and the tableau algorithm.

Semantic Characterisation: Direct Attempt

S4$^{\mathsf{u}}$ is the extension of S4 with the universal modality $[\mathsf{u}]$:

$$\mathcal{M}, x \models [\mathsf{u}]\phi \text{ iff } \mathcal{M}, y \models \phi \text{ for all } y.$$

Theorem 30

α/β is derivable in S4 iff $[\mathsf{u}]\alpha \wedge \neg\beta$ is unsatisfiable in S4^{u}.

Theorem 31

α/β is admissible for S4 iff $[\mathsf{u}]\alpha \wedge \neg\beta$ is unsatisfiable in the extension of S4^{u} which is characterized by the models with co-cover property.

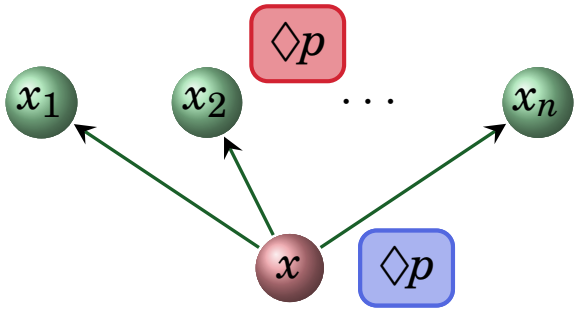
Problem: the filtration argument for the finite model property fails!

Semantic Characterisation I

- *Formula Definable Co-Cover Property*:

(FCCP) $\exists x \forall p \, (\mathcal{M}, x \models \Diamond p \rightarrow \mathcal{M}, x \models p)$

$$\forall x_1 \cdots \forall x_n \exists x \forall p \, \big(R(x, x_1) \wedge \cdots \wedge R(x, x_n) \wedge$$
$$\mathcal{M}, x \models \Diamond p \rightarrow (\mathcal{M}, x \models p \vee \mathcal{M}, x_1 \models \Diamond p \vee \cdots \vee \mathcal{M}, x_n \models \Diamond p) \big).$$



- $S4^{u,+}$ is characterised by the class $S4^{u}$-models with (FCCP).

Semantic Characterisation II

Theorem 32 (The Effective Finite Model Property)

Let ϕ be a formula and n the length of ϕ. If ϕ is satisfiable in an $\mathsf{S4}^{u,+}$-model then it is satisfiable in a finite $\mathsf{S4}^{u,+}$-model and its size does not exceed 2^n.

Theorem 33

α/β is admissible for $\mathsf{S4}$ iff $[\mathrm{u}]\alpha \wedge \neg\beta$ is unsatisfiable in $\mathsf{S4}^{u,+}$.

Semantic Specification of S4$^{\mathsf{u}}$

- S^0 = equivalences defining connectives.

  universal modality: $\forall x\ (\nu_1([\mathsf{u}]p, x) \leftrightarrow \forall y\ \nu_1(p, y))$.

- S^b = background theory: all logical terms are atomic $\supseteq$ frame conditions.

  reflexivity: $\forall x\, R(x, x)$,

  transitivity: $\forall x \forall y \forall z\ (R(x, y) \wedge R(y, z) \rightarrow R(x, z))$.

- S^b contains the usual equality axioms.

Co-Cover Property Specification

(FCCP) $$\exists x \forall p \, (\nu_1(\Diamond p, x) \rightarrow \nu_1(p, x))$$

$$\forall x_1 \cdots \forall x_n \exists x \forall p \, \big(R(x, x_1) \wedge \cdots \wedge R(x, x_n) \wedge$$

$$\nu_1(\Diamond p, x) \rightarrow (\nu_1(p, x) \vee \nu_1(\Diamond p, x_1) \vee \cdots \vee \nu_1(\Diamond p, x_n))\big)$$

$$\exists x \forall p \forall y \, \big((R(x, y) \wedge \nu(p, y)) \rightarrow \nu(p, x)\big)$$

$$\forall x_1 \cdots \forall x_n \exists x \forall p \forall y \exists z \, \big(R(x, x_1) \wedge \cdots \wedge R(x, x_n) \wedge$$

$$(R(x, y) \wedge \nu_1(p, y) \rightarrow \big(\nu_1(p, x) \vee (\nu_1(p, z) \wedge (R(x_1, z) \vee \cdots \vee R(x_n, z)))\big)\big)$$

(FCCP′) $$\forall y \, \big((R(g_0, y) \wedge \nu(p, y)) \rightarrow \nu(p, g_0)\big)$$

$$\forall x_1 \cdots \forall x_n \big(R(g_n(x_1, \ldots, x_n), x_1) \wedge \cdots \wedge R(g_n(x_1, \ldots, x_n), x_n)\big)$$

$$\forall x_1 \cdots \forall x_n \forall y \exists z \, \big((R(g_n(x_1, \ldots, x_n), y) \wedge \nu_1(p, y)) \rightarrow$$

$$\big(\nu_1(p, g_n(x_1, \ldots, x_n)) \vee (\nu_1(p, z) \wedge (R(x_1, z) \vee \cdots \vee R(x_n, z)))\big)\big)$$

g_n = Skolem symbols

Generated Rules for $S4^{u,+}$

Rules for the co-cover property

$$(\mathrm{cc}^0)\colon \frac{}{\neg R(g_0,y) \mid \neg\nu(p,y) \mid \nu(p,g_0)}$$

$$(\mathrm{cc}_0^n)\colon \frac{}{R(g_n(\overline{x}),x_1),\ \ldots,\ R(g_n(\overline{x}),x_n)}$$

$$(\mathrm{cc}_1^n)\colon \frac{}{\begin{array}{l}\neg R(g_n(\overline{x}),y) \mid \neg\nu(p,y) \mid \nu(p,g_n(\overline{x})) \mid \\ \mid R(x_1,h_n(p,\overline{x},y)),\ \ \nu(p,h_n(p,\overline{x},y)) \mid \cdots\end{array}}$$

Refined tableau calculus Tab^{+} for $S4^{u,+}$

Decomposition tableau rules

$$\frac{@_i \neg\neg p}{@_i p}$$

$$\frac{@_i (p \vee q)}{@_i p \mid @_i q} \qquad \frac{@_i \neg (p \vee q)}{@_i \neg p,\ @_i \neg q}$$

$$\frac{@_i \neg \Box p}{@_i \Diamond\{f(p,i)\},\ @_{f(p,i)} \neg p} \qquad \frac{@_i \Box p,\ @_i \Diamond\{j\}}{@_j p}$$

$$\frac{@_i \neg [\mathsf{u}] p}{@_{f_{\mathsf{u}}(p)} \neg p} \qquad \frac{@_i [\mathsf{u}] p}{@_j p}$$

Theory tableau rules

$$\frac{}{@_i \Diamond\{i\}} \qquad \frac{@_i \Diamond\{j\},\ @_j \Diamond\{k\}}{@_i \Diamond\{k\}}$$

(FCCP′) tableau rules ($n > 0$)

$$\frac{}{@_{g_0} \{g_0\}} \qquad \frac{@_{g_0} \Diamond\{i\},\ @_i p}{@_{g_0} p} \qquad \frac{}{@_{g_n(\bar{i})} \Diamond\{i_1\},\ \ldots,\ @_{g_n(\bar{i})} \Diamond\{i_n\}}$$

$$\frac{@_{g_n(\bar{i})} \Diamond\{j\},\ @_j p}{@_{g_n(\bar{i})} p \mid @_{i_1} \Diamond\{h_n(p,\bar{i},j)\},\ @_{h_n(p,\bar{i},j)} p \mid \cdots \mid @_{i_n} \Diamond\{h_n(p,\bar{i},j)\},\ @_{h_n(p,\bar{i},j)} p}$$

Tableau Calculi for $S4^u$ and $S4^{u,+}$

- $Tab_{S4^{u,+}} \stackrel{\text{def}}{=} Tab^+ + (\text{ub})$.
- $Tab_{S4^u} \stackrel{\text{def}}{=} Tab_{S4^{u,+}} - \{(\text{FCCP}')\text{-rules}\}$.
- Tab_{S4^u} is a reformulation of the usual calculus for hybrid $S4^u$.

Theorem 34

Tab_{S4^u} and $Tab_{S4^{u,+}}$ are sound, (constructively) complete, and terminating tableau calculi for $S4^u$ and $S4^{u,+}$ respectively.

The Tableau Algorithm

Step 1. Rewrite α/β to $[\mathsf{u}]\alpha \wedge \neg\beta$.

Step 2. Run $\mathit{Tab}_{\mathsf{S4u}}$ on $[\mathsf{u}]\alpha \wedge \neg\beta$.

Step 3. If $\mathit{Tab}_{\mathsf{S4u}}([\mathsf{u}]\alpha \wedge \neg\beta)$ is closed then return **'derivable'**.

Step 4. Otherwise *continue* the derivation in $\mathit{Tab}_{\mathsf{S4u,+}}$.

Step 5. If $\mathit{Tab}_{\mathsf{S4u,+}}([\mathsf{u}]\alpha \wedge \neg\beta)$ is closed then return **'not derivable but admissible'**. Otherwise return **'not admissible'**.

MetTeL and The Framework

- The framework: a semi-automatic way of generating tableau calculi.
- Is there a possibility to generate a prover from a logic specification?
- Yes: we need a system which can accept tableau calculi as a parameter.
- MetTeL implements this feature.

Objectives behind the MetTeL Implementation

- Experiments with different tableau rules for new logics.
- Quick implementation of decision procedures.
- A prover which can handle frequent changes in calculi.
- A prover which runs relatively fast.

What is MetTeL?

- MetTeL has a logic-independent inference engine

- Closely related systems are The Tableau Work Bench and LoTREC.

Simple Example I

- Input sets: $\{P \wedge Q, \neg P\}$ and $\{P \wedge Q\}$.
- Tableau calculus:

$$\frac{@_i P, \quad @_i \neg P}{\bot} \qquad \frac{@_i \neg\neg P}{@_i P}$$

$$\frac{@_i (P \wedge Q)}{@_i P, \quad @_i Q} \qquad \frac{@_i \neg (P \wedge Q)}{@_i \neg P \mid @_i \neg Q}$$

- Prepare a file **`simple.tbl`** with the content:

```
@i P  @i ~P / {FALSE};   @i ~~P / {@i P};
@i (P&Q) / {@i P @i Q};  @i ~(P&Q) / {@i ~P} | { @i ~Q};
```

Simple Example II

- Run MetTeL:

```
>java -jar mettel.jar -tbl simple.tbl
```

- Enter the problem set in the prompt:

```
P & Q     ~P
```

- The output will be

```
Unsatisfiable
```

- If input is **P & Q** then the output is

```
Satisfiable
MODEL:[(@{n0}Q), (@{n0}P), (@{n0}(~((~P)|(~Q))))]
Compact representation:[(@{n0}Q), (@{n0}P)]
```

Predefined Tableau Calculi

`>java -jar mettel.jar [<predefined-tableau>] [-i <in-file>]`

`<predefined-tableau>` is one of the following:

`-bool`	Classical propositional logic.
`-hl`	Hybrid logic HL(@) with relational union and composition (Default).
`-hlu`	Hybrid logic HL(@,u) with the universal modality.
`-met`	Metric logic (without the ‘closer’ operator).
`-topo`	Metric logic with the topology operator.
`-albo`	Description logic $\mathcal{ALBO}$ with full role negation operator.
`-alboid`	Description logic $\mathcal{ALBO}^{\mathrm{id}}$ with full role negation operator and the identity role.

`<in-file>` — name of the file with the problem set.

Logical Syntax

Formulae

Booleans

$\bot$	$\top$	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
FALSE	**TRUE**	**~P**	**P&Q**	**P|Q**	**P->Q**	**P<->Q**

Modal

‘Box’	‘Diamond’	Universal modality	‘Somewhere’
$[R]P$	$\langle R \rangle P$	$[\mathrm{u}]P$	$\langle \mathrm{u} \rangle P$
forall{R}P	**exists{R}P**	**forall P**	**exists P**

Hybrid ‘at’ operator

$@_i P$	**@i P**

Logical Syntax

Relations

Complement (role negation)	$\neg R$	**~R**
Inverse	R^{-1}	**R-**
Reflexive-transitive closure	R^*	**R***
Intersection	$R \cap S$	**R&S**
Union	$R \cup S$	**R|S**
Composition	$R \circ S$	**R;S**

Tableau Rule

- General template:

$$\frac{X_0}{X_1 \mid \cdots \mid X_n},$$

  where $n \geq 0$ and X_i are sets of formulae.

- MetTeL representation:

  <formulae>/{<formulae>} | ... | {<formulae>},

  where **<formulae>** denotes a list of formulae.

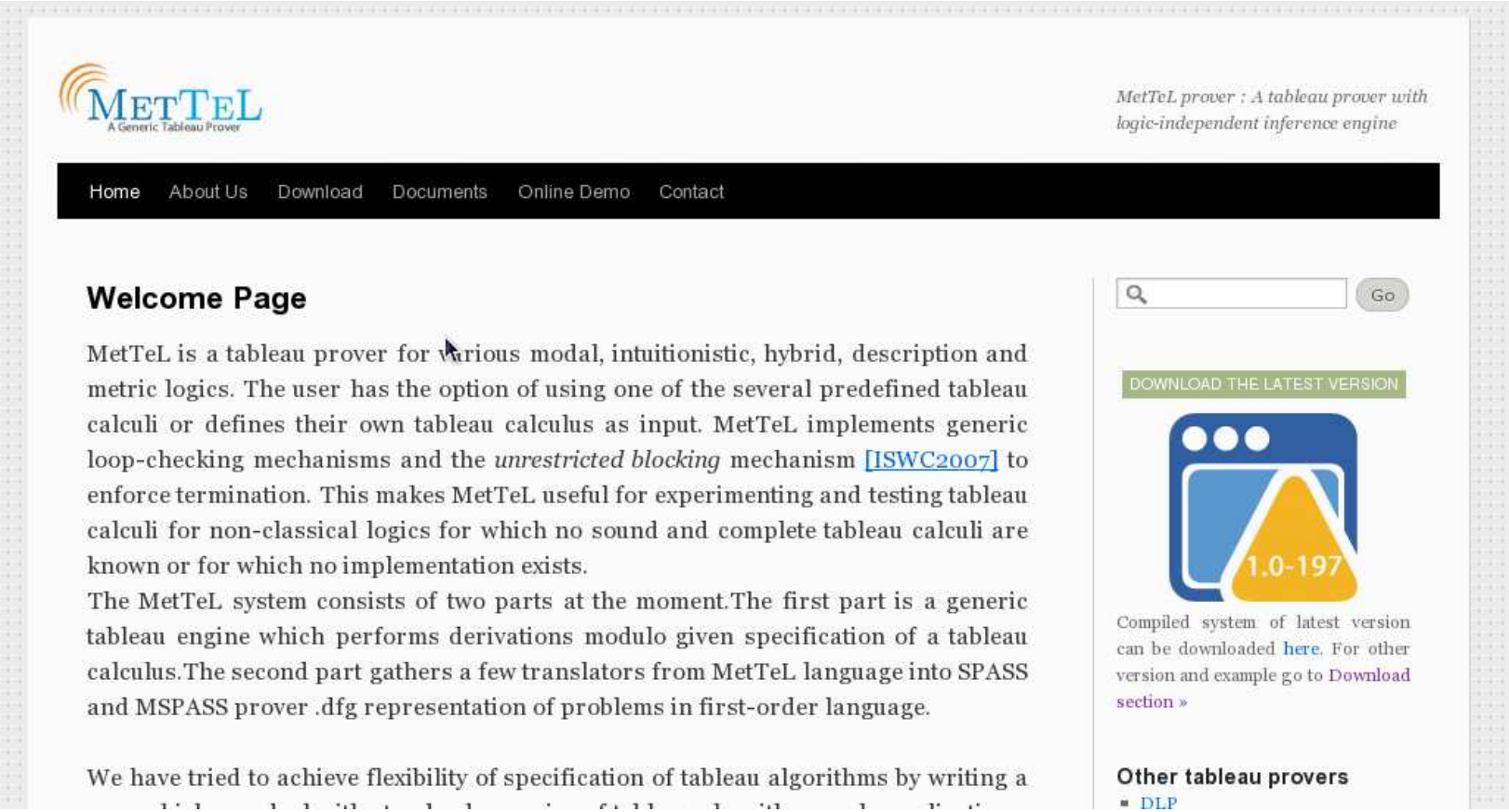
- Clash rule: **<formulae>/{FALSE}**

MetTeL Distinctive Features

- Simple input language.
- Use of Skolem terms in premises of tableau rules.
- Two generic loop-checking mechanisms:
  - Generalisation of anywhere equality blocking.
  - Unrestricted blocking rule mechanism.
- MetTeL is the only prover which can handle logics with full negation of roles such as $\mathcal{ALBO}^{\mathrm{id}}$ and its sublogics.
- MetTeL is still the only prover which decides the logic of metric and topology.

Welcome to MetTeL Web site!

http://www.mettel-prover.org



Online Demo is available.

Concluding remarks

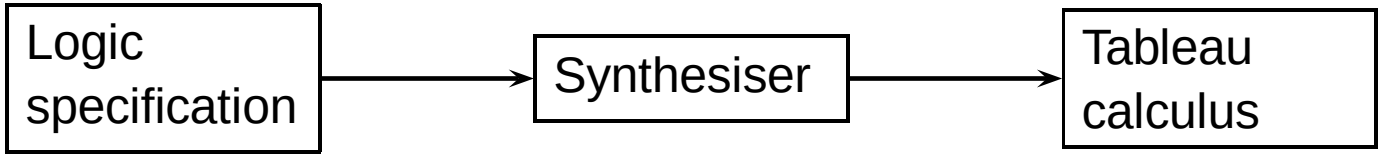
The tableau calculus synthesis framework:

- provides a method for turning FO semantic specifications into
  - sound and complete tableau calculi, and
  - decision procedures, when possible
- covers many modal and description logics

MetTeL, along with LoTReC or the Tableau Workbench, can be used as provers for the generated tableau calculi

Our vision

- Automatic tableau calculus synthesiser



- Extend and improve the framework
  - with more refinements
  - with notion of redundancy
  - to generate other kinds of tableau calculi, other types of deduction calculi . . .
  - to cover non-first-order definable logics
  - weaken some of the conditions

References

Tableau methods:

- D'Agostino, M., Gabbay, D., Hähnle, R. and Posegga, J. (1999), *Handbook of Tableau Methods*. Kluwer.

Tableau calculus synthesis:

- Schmidt, R. A. and Tishkovsky, D. (2011), Automated Synthesis of Tableau Calculi. *Logical Methods in Computer Science* **7** (2), 1–32. Short version published in *Proc. TABLEAUX 2009*, LNCS 5607, Springer, 310–324 (2009).
- Babenyshev, S., Rybakov, V., Schmidt, R. A., and Tishkovsky, D. (2010), A Tableau Method for Checking Rule Admissibility in S4. *ENTCS* **262**, 17–32.
- Tishkovsky, D., Schmidt, R. A., and Khodadadi, M. (2011), MetTeL: A Tableau Prover with Logic-Independent Inference Engine. In *Proc. TABLEAUX 2011*, LNCS 6793, Springer, 242–247.

Terminating tableau calculi for description logics with role negation:

- Schmidt, R. A., and Tishkovsky, D. (2007), Using Tableau to Decide Expressive Description Logics with Role Negation. In *Proc. ISWC 2007 + ASWC 2007*. LNCS 4825, Springer, 438–451.
- Schmidt, R. A., and Tishkovsky, D. (2008), A General Tableau Method for Deciding Description Logics, Modal Logics and Related First-Order Fragments. In *Automated Reasoning (IJCAR 2008)*. LNCS 5195, Springer, 194–209.

Eliminating $\exists$ quantifiers using Skolemisation

Intuition: Skolemisation replaces $\exists y$ by a (choice) function f that computes y from all the arguments y depends on

Skolemisation can be performed by using this rewrite rule:

$$\forall x_1 \ldots \forall x_n \exists y\, F \quad \Rightarrow_{\mathrm{Sk}} \quad \forall x_1 \ldots \forall x_n\, F\{y/f(x_1, \ldots, x_n)\}$$

where f is a fresh function symbol

- Always apply outermost first, *not* in subformulae
- f is called a *Skolem function*;
  $f(x_1, \ldots, x_n)$ is called a *Skolem term*, or *Skolem constant* when $n = 0$

- Example:

$$\forall x \exists y\, P(x, y) \quad \Rightarrow_{\mathrm{Sk}} \quad \forall x\, P(x, f(x))$$

Theorem 35 (Skolemisation does not affect (un)satisfiability)

If $F \Rightarrow^*_{\mathrm{Sk}} G$ *then* F *is satisfiable iff* G *is satisfiable*