Generating Tableau Provers Using METTEL^2

Tutorial

Dmitry Tishkovsky and Renate Schmidt

School of Computer Science
The University of Manchester

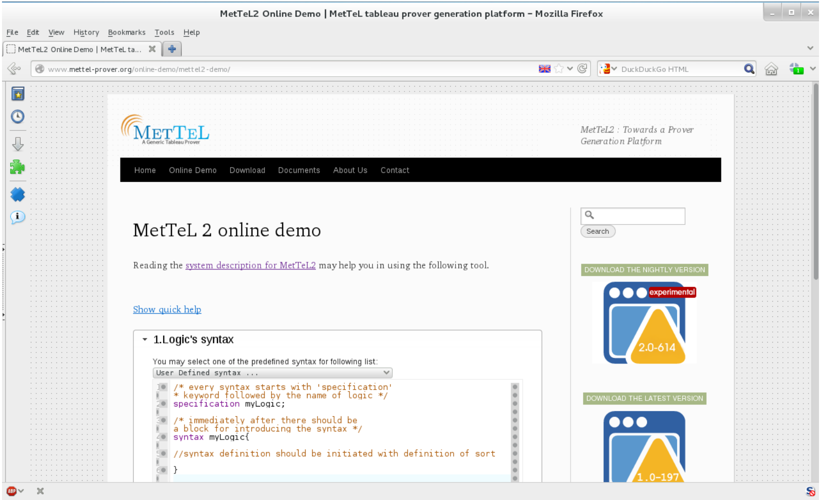
THE 22ND INTERNATIONAL CONFERENCE
ON AUTOMATED REASONING WITH ANALYTIC TABLEAUX AND RELATED METHODS

16 – 19 SEPTEMBER 2013

Outline

Outline

http://www.mettel-prover.org



Outline

Prover Development Problem

- Different applications require different (logical) formalisms
- Logics need reasoning tools
- Implementation of provers is expensive
- Altering existing provers is hard
- Translational approach requires additional knowledge and skills for the user

Solution

Generation of a prover code from a specification of a logic.

Goals and Objectives of Implementation of METTEL^2

- An easy to use prover generator
- Modularity of generated code
- Hierarchy of public JAVA classes and interfaces that can be easily extended and integrated with other systems

Outline

Requirements and Installation

Requires JAVA. Tested on JRE 1.6 and 1.7.

Installation

- Download and install JRE 1.7 from `http://www.java.com`.
- Download **`mettel-2.0-XXX.zip`** from `http://www.mettel-prover.org`.
- Place all the files from **`mettel-2.0-XXX.zip`** into some directory, e.g. **`~/mettel2`**.

MANCHESTER 1824 The University of Manchester

Distribution Kit and Library Dependencies

mettel-2.0-XXX.zip

- **`mettel2.jar`** — main **`jar`**-file for the generator, includes all the packages, depends on the ANTLR libraries:
  - **`antlr.jar`**,
  - **`antlr3.jar`**,
  - **`antlr3-runtime.jar`**,
  - **`stringtemplate4.jar`**,

  and the CSV formatting library:
  - **`opencsv.jar`** — required for running benchmark suites.
- **`mettel2-core.jar`** — required for running the generated provers, depends on:
  - **`antlr3-runtime.jar`** — required for running parsers generated by ANTLR.
- **`mettel2-util.jar`** — required for running expression generators and benchmark suites.

Outline

Generating and Running Prover

- Prepare a specification file **S4.s**.
- Generate the prover:

```
> java -jar ~/mettel2/mettel2.jar -i S4.s
```

- Place the generated **S4.jar** into **~/mettel2**.
- Run the prover:

```
> java -jar ~/mettel2/S4.jar
```

Package Structure of Generated Provers

S4.jar

- Syntax related classes: **`S4.language.S4`**
- Tableau related classes: **`S4.tableau.S4`**
- Executable classes:
  - Main class for the prover: **`S4.tableau.S4.S4TableauProver`**
  - Main class for running benchmark suites: **`S4.tableau.S4.S4Benchmark`**
  - Generator of random expressions for specified syntax: **`S4.language.S4.util.S4RandomExpressionGenerator`**
  - Statistical analyser of expressions: **`S4.language.S4.S4ProblemAnalyzer`**
- Additional resources and examples: **`etc`**

Outline

Specification

Short syntax

```
specification <name>;
options{
...
}
syntax <name>{
...
}
tableau <name>{
...
}
```

Full syntax

```
specification <path>;
options{
...
}
syntax <name>
options{
...
}{
...
}
tableau <name> in syntax <name>
options{
...
}{
...
}
<other syntax and tableau blocks>
```

Inside Syntax Specification

- Declaration of sorts of a (multi-sorted) propositional language:

```
sort formula, world;
```

- Connective specifications as simple BNF statements:

```
formula at = '@' world formula;
formula disjunction = formula '|' formula;
```

Formula examples

P|~Q, @l P,
@l(P|~Q),
@l ~(p|(P|~Q)),
@l ~p|P|~Q

$P \vee \neg Q$, $@_\ell P$,
$@_\ell(P \vee \neg Q)$,
$@_\ell \neg(p \vee (P \vee \neg Q))$,
$@_\ell \neg p \vee P \vee \neg Q$

Example

$$\phi \stackrel{\text{def}}{=} p \mid \bot \mid \neg\phi \mid \Diamond\phi \mid @_w\phi \mid \phi \vee \phi \mid w \approx w \mid R(w, w)$$

$$w \stackrel{\text{def}}{=} i \mid f(w, \phi)$$

```
syntax S4{
    sort formula, world;
    formula false = 'false';
    formula negation = '~' formula;
    formula diamond = '<>' formula;
    formula at = '@' world formula;
    formula disjunction = formula '|' formula;
    formula equality = '[' world '=' world ']'; //Equality
    formula relation = 'R' '(' world ',' world ')'; //Relation
    world f = 'f' '(' world ',' formula ')'; //Skolem function
}
```

Inside Tableau Specification

- Tableau rule declaration in a premise-conclusion syntax:

```
@l (P | Q) / @l P $| @l Q $;
```

$$\frac{@_\ell (P \vee Q)}{@_\ell P \mid @_\ell Q}$$

- Branch separator $| and rule separator $;
- Rule priorities:

```
@l (P | Q) / @l P $| @l Q priority 2 $;
```

- Default priority is 0

Options for Tableau Specification

- Redefinable separators $| and $;

```
options{
...
tableau.rule.delimiter =;
tableau.rule.branch.delimiter=||
}
...
tableau S4{
  ...
  @I (P | Q)  /  @I P ||  @I Q priority 3;
}
```

- The delimiter / between premises and conclusions can be redefined via **tableau**.rule.premise.delimiter property.
- Must be redefined in the global **options** block or in the **options** block for the corresponding syntax.

Example

```
tableau S4{
    @i ~(~P) / @i P priority 1; //Double−negation removal
    @i(P|Q) / @i P || @i Q priority 3; //Disjunction rule
    @i~(P|Q) / @i~P @i~Q priority 1; //"Conjunction" rule
    @i<>P / R(i,f(i,P)) @f(i,P)P priority 7; //Diamond rule
    @i~(<>P) R(i,j) / @j~P priority 2; //"Box" propagation rule
    @i P / R(i,i) priority 1; //Reflexivity
    R(i,j) R(j,k) / R(i,k) priority 2; //Transitivity
    @i P @i~P / priority 0; //Closure rule
    R(i,j) / [i=j] || ~([i=j]) priority 6; //Ancestor blocking rule
    ~([i=i])/ priority 0; //Closure rule for inequality
}
```

Notice separators in tableau rules!

Options

```
options{
    branch.bound=
    branch.selection.strategy=

    equality.keywords={equality}

    tableau.rule.delimiter=$;
    tableau.rule.branch.delimiter=$|
    tableau.rule.premise.delimiter=/
}
```

Outline

Features of Generated Provers

- Dynamic backtracking
- Conflict directed backjumping
- DFLR or BF search strategy
- Rule applications are controlled via rule priorities
- Equality reasoning via backward and forward rewriting

DFLR and BF Search Strategies

- Option for depth-first left-to-right search strategy:

```
branch.selection.strategy = \
  mettel.core.tableau.MettelSimpleLIFOBranchSelectionStrategy
```

- Some logics, e.g. $\mathcal{ALBO}^{\mathrm{id}}$, require fair branch selection strategy for termination.
- Option for breadth-first search strategy:

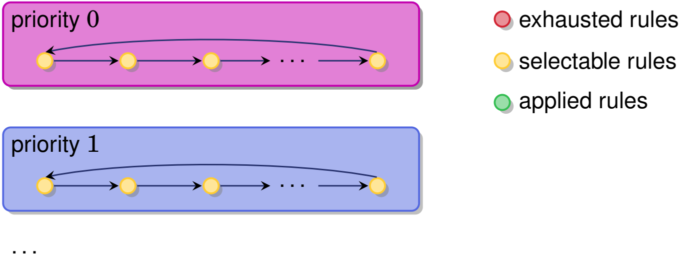
```
branch.selection.strategy = \
  mettel.core.tableau.MettelSimpleFIFOBranchSelectionStrategy
```

- Other implementations of the interface MettelBranchSelectionStrategy are allowed.

Generic Protocol for Rule Priorities

- A rule can be selected only if the queue of expressions associated with the rule is not empty (rule is *selectable*).
- Selection must be fair within each priority group: if a rule is selectable then it will be selected eventually.
- A rule can be selected only if all rules with smaller priority values (higher priority) are not selectable.
- Note that selectable rule is not necessarily applicable.

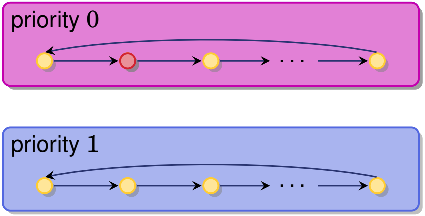
Implemented Rule Selection Algorithm



Example

In order to achieve that the closure rule is applied immediately after any new information is added to a branch assign to the closure rule the priority value 0 and to all other rule values higher than 0, e.g., 1.

Implemented Rule Selection Algorithm



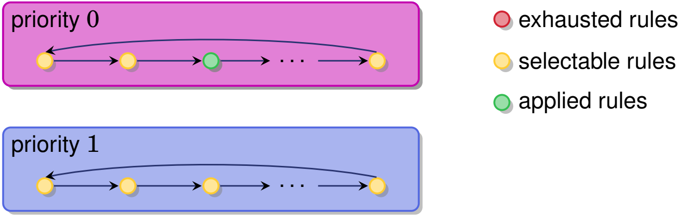
- exhausted rules
- selectable rules
- applied rules

. . .

Example

In order to achieve that the closure rule is applied immediately after any new information is added to a branch assign to the closure rule the priority value 0 and to all other rule values higher than 0, e.g., 1.

Implemented Rule Selection Algorithm

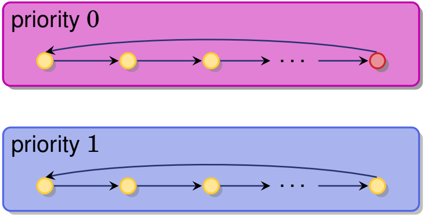


. . .

Example

In order to achieve that the closure rule is applied immediately after any new information is added to a branch assign to the closure rule the priority value 0 and to all other rule values higher than 0, e.g., 1.

Implemented Rule Selection Algorithm



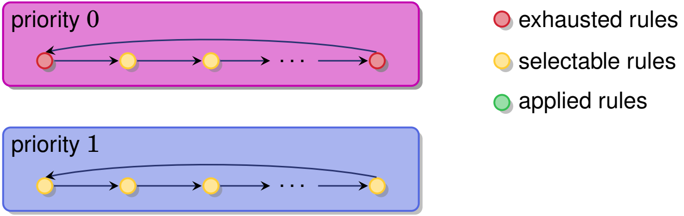
- exhausted rules
- selectable rules
- applied rules

. . .

Example

In order to achieve that the closure rule is applied immediately after any new information is added to a branch assign to the closure rule the priority value 0 and to all other rule values higher than 0, e.g., 1.

Implemented Rule Selection Algorithm

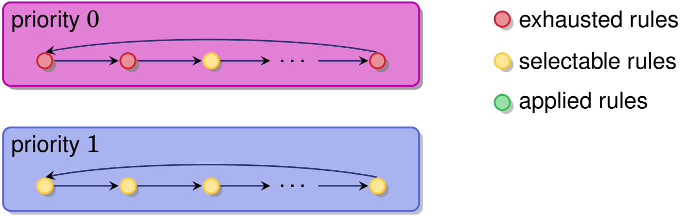


. . .

Example

In order to achieve that the closure rule is applied immediately after any new information is added to a branch assign to the closure rule the priority value 0 and to all other rule values higher than 0, e.g., 1.

Implemented Rule Selection Algorithm

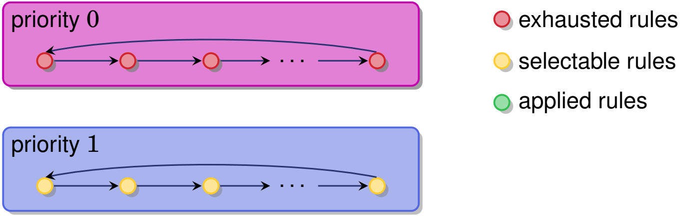


. . .

Example

In order to achieve that the closure rule is applied immediately after any new information is added to a branch assign to the closure rule the priority value 0 and to all other rule values higher than 0, e.g., 1.

Implemented Rule Selection Algorithm

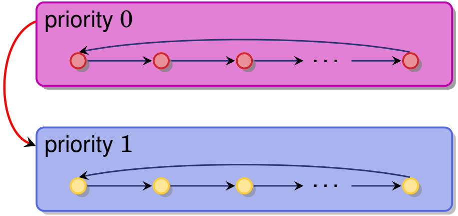


...

Example

In order to achieve that the closure rule is applied immediately after any new information is added to a branch assign to the closure rule the priority value 0 and to all other rule values higher than 0, e.g., 1.

Implemented Rule Selection Algorithm



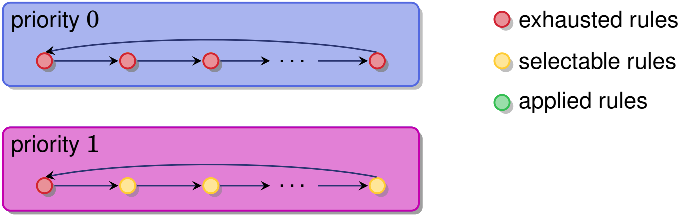
. . .

- exhausted rules
- selectable rules
- applied rules

Example

In order to achieve that the closure rule is applied immediately after any new information is added to a branch assign to the closure rule the priority value 0 and to all other rule values higher than 0, e.g., 1.

Implemented Rule Selection Algorithm

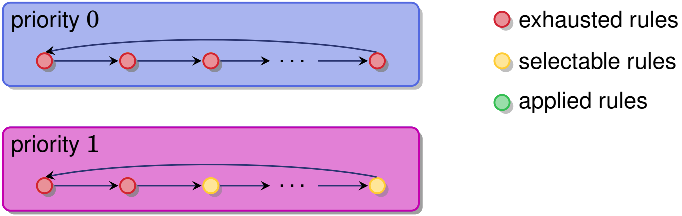


. . .

Example

In order to achieve that the closure rule is applied immediately after any new information is added to a branch assign to the closure rule the priority value 0 and to all other rule values higher than 0, e.g., 1.

Implemented Rule Selection Algorithm

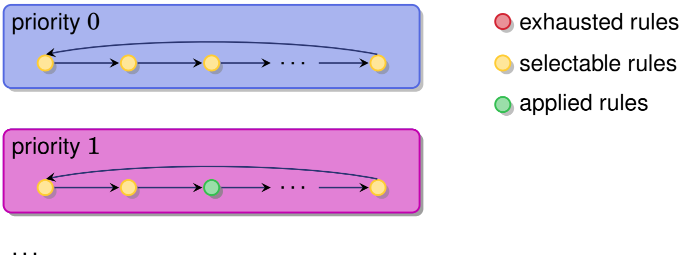


. . .

Example

In order to achieve that the closure rule is applied immediately after any new information is added to a branch assign to the closure rule the priority value 0 and to all other rule values higher than 0, e.g., 1.

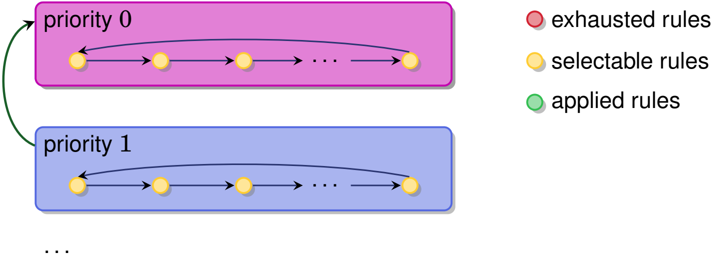
Implemented Rule Selection Algorithm



Example

In order to achieve that the closure rule is applied immediately after any new information is added to a branch assign to the closure rule the priority value 0 and to all other rule values higher than 0, e.g., 1.

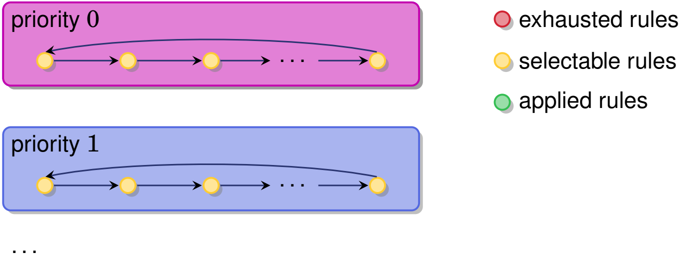
Implemented Rule Selection Algorithm



Example

In order to achieve that the closure rule is applied immediately after any new information is added to a branch assign to the closure rule the priority value 0 and to all other rule values higher than 0, e.g., 1.

Implemented Rule Selection Algorithm



Example

In order to achieve that the closure rule is applied immediately after any new information is added to a branch assign to the closure rule the priority value 0 and to all other rule values higher than 0, e.g., 1.

A Guideline for Choosing Rule Priorities

Given two rules ρ and ρ', set priority value of ρ less than the priority value of ρ' if

- ρ has lower branching factor than ρ', or
- branching factors of ρ and ρ' are equal but ρ has less premises.

Additional optimisation is possible, if each set of expressions has a complexity measure.

Example

```
tableau S4{
    @i ~(~P) / @i P priority 1; //Double−negation removal
    @i(P|Q) / @i P || @i Q priority 3; //Disjunction rule
    @i~(P|Q) / @i~P @i~Q priority 1; //"Conjunction" rule
    @i<>P / R(i,f(i,P)) @f(i,P)P priority 7; //Diamond rule
    @i~(<>P) R(i,j) / @j~P priority 2; //"Box" propagation rule
    @i P / R(i,i) priority 1; //Reflexivity
    R(i,j) R(j,k) / R(i,k) priority 2; //Transitivity
    @i P @i~P / priority 0; //Closure rule
    R(i,j) / [i=j] || ~([i=j]) priority 6; //Ancestor blocking rule
    ~([i=i])/ priority 0; //Closure rule for inequality
}
```

Outline

Why Rewriting?

- Considerably simplifies derivations:
  - Exhaustive rewriting:

$$\text{if } f(i,P) \stackrel{\mathcal{R}}{\rightarrow} i \text{ then } f(f(f(i,P),P),P) \stackrel{\mathcal{R}}{\rightarrow} i.$$

  - On-the-fly simplification — forward rewriting.
- Rewrite system can be efficiently maintained.
- Equality reasoning is required for the generic blocking mechanism.

MANCHESTER 1824 The University of Manchester

Equality Reasoning via Forward and Backward Rewriting

- A lexicographic path ordering $\prec$ on all expressions of current tableau branch.
- Rewrite relation $\mathcal{R}$.
- Forward rewriting: every new expression in a branch is rewritten wrt $\mathcal{R}$.
- Backward rewriting is triggered by equality expressions:
  - every equality expression $\boldsymbol{E}(\alpha, \beta)$ in current branch is oriented wrt $\prec$, and
  - either $\alpha \stackrel{\mathcal{R}}{\to} \beta$ or $\beta \stackrel{\mathcal{R}}{\to} \alpha$ is added to $\mathcal{R}$, and
  - all the expressions in the branch are rewritten wrt $\mathcal{R}$.

Specifying Equality Expressions

- Option equality.keywords:

Default: equality.keywords = {equality}

```
options{
equality.keywords = {equality, equivalence}
}
```

- Corresponding name in a binary connective specification:

```
syntax SomeLogic{
  ...
  formula equality = '[' nominal '=' nominal ']';
  formula equivalence = formula '<->' formula;
}
```

Outline

MANCHESTER 1824 The University of Manchester

Blocking Mechanisms

- There are periodicities in tableau derivations.
- They are usually detected with some form of loop checking mechanisms:
  - subset or equality blocking,
  - ancestor or anywhere blocking,
  - static or dynamic blocking,
  - pattern-based blocking, etc.
- The standard loop-checking mechanisms are tied to particular logics.
- In METTEL2, a generic blocking mechanism, called unrestricted blocking mechanism, can be specified.

Unrestricted Blocking Rule

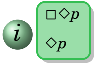
$$\text{(ub):}\ \frac{}{x \approx y \mid x \not\approx y}$$

Termination Condition

In every open branch there is some node from which point onwards, all possible applications of the (ub) rule have been performed before any application of any term-generating rule.

- Early blocking:

  Standard loop checking:



- Relies on tableau backtracking
- Provides termination for logics with FMP

Unrestricted Blocking Rule

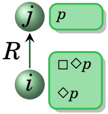
$$\text{(ub):}\ \frac{}{x \approx y \mid x \not\approx y}$$

Termination Condition

In every open branch there is some node from which point onwards, all possible applications of the (ub) rule have been performed before any application of any term-generating rule.

- Early blocking:

Standard loop checking:



- Relies on tableau backtracking
- Provides termination for logics with FMP

Unrestricted Blocking Rule

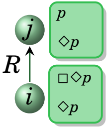
$$\text{(ub): } \frac{}{x \approx y \mid x \not\approx y}$$

Termination Condition

In every open branch there is some node from which point onwards, all possible applications of the (ub) rule have been performed before any application of any term-generating rule.

- Early blocking:

  Standard loop checking:



- Relies on tableau backtracking
- Provides termination for logics with FMP

Unrestricted Blocking Rule

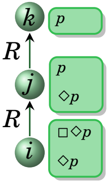
$$\text{(ub):}\ \frac{}{x \approx y \mid x \not\approx y}$$

Termination Condition

In every open branch there is some node from which point onwards, all possible applications of the (ub) rule have been performed before any application of any term-generating rule.

- Early blocking:

Standard loop checking:



- Relies on tableau backtracking
- Provides termination for logics with FMP

Unrestricted Blocking Rule

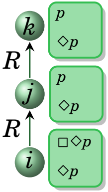
(ub): $\dfrac{}{x \approx y \mid x \not\approx y}$

Termination Condition

In every open branch there is some node from which point onwards, all possible applications of the (ub) rule have been performed before any application of any term-generating rule.

- Early blocking:

  Standard loop checking:



- Relies on tableau backtracking
- Provides termination for logics with FMP

Unrestricted Blocking Rule

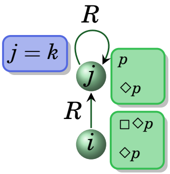
$$\text{(ub)}: \frac{}{x \approx y \mid x \not\approx y}$$

Termination Condition

In every open branch there is some node from which point onwards, all possible applications of the (ub) rule have been performed before any application of any term-generating rule.

- Early blocking:

Standard loop checking:



- Relies on tableau backtracking
- Provides termination for logics with FMP

Unrestricted Blocking Rule

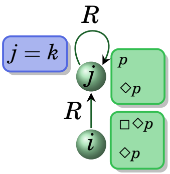
$$\text{(ub):}\ \frac{}{x \approx y \mid x \not\approx y}$$

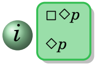
Termination Condition

In every open branch there is some node from which point onwards, all possible applications of the (ub) rule have been performed before any application of any term-generating rule.

- Early blocking:

Standard loop checking: (ub):





- Relies on tableau backtracking
- Provides termination for logics with FMP

Unrestricted Blocking Rule

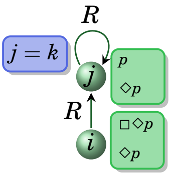
(ub): $\dfrac{}{x \approx y \mid x \not\approx y}$

Termination Condition

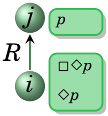
In every open branch there is some node from which point onwards, all possible applications of the (ub) rule have been performed before any application of any term-generating rule.

- Early blocking:

Standard loop checking:



(ub):



- Relies on tableau backtracking
- Provides termination for logics with FMP

Unrestricted Blocking Rule

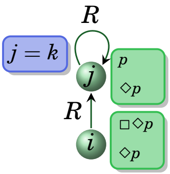
(ub): $\dfrac{}{x \approx y \mid x \not\approx y}$

Termination Condition

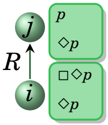
In every open branch there is some node from which point onwards, all possible applications of the (ub) rule have been performed before any application of any term-generating rule.

- Early blocking:

Standard loop checking:



(ub):



- Relies on tableau backtracking
- Provides termination for logics with FMP

Unrestricted Blocking Rule

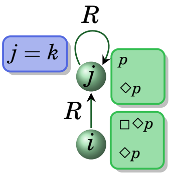
(ub): $\dfrac{}{x \approx y \mid x \not\approx y}$

Termination Condition

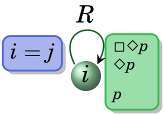
In every open branch there is some node from which point onwards, all possible applications of the (ub) rule have been performed before any application of any term-generating rule.

- Early blocking:

Standard loop checking:



(ub):



- Relies on tableau backtracking
- Provides termination for logics with FMP

Specifying the Unrestricted Blocking Mechanism

- Ensure that equality reasoning is turned on.
- Append equality expressions to the syntax specification.
- Specify the unrestricted blocking rule:

```
tableau S4{
  ...
  @i P @j Q / [i=j] || ~[i=j];
}
```

- Satisfy the termination condition by using rule priorities:

```
tableau S4{
  ...
  @i<>P / R(i,f(i,P)) @f(i,P)P priority 7; //Diamond rule
  ...
  @i P @j Q / [i=j] || ~([i=j]) priority 6; //Blocking rule
}
```

Example

```
syntax S4{
    sort formula, world;
    formula false = 'false';
    formula negation = '~' formula;
    formula diamond = '<>' formula;
    formula at = '@' world formula;
    formula disjunction = formula '|' formula;
    formula equality = '[' world '=' world ']'; //Equality
    formula relation = 'R' '(' world ',' world ')'; //Relation
    world f = 'f' '(' world ',' formula ')'; //Skolem function
}
tableau S4{
    @i ~(~P) / @i P priority 1; //Double−negation removal
    @i(P|Q) / @i P || @i Q priority 3; //Disjunction rule
    @i~(P|Q) / @i~P @i~Q priority 1; //"Conjunction" rule
    @i<>P / R(i,f(i,P)) @f(i,P)P priority 7; //Diamond rule
    @i~(<>P) R(i,j) / @j~P priority 2; //"Box" propagation rule
    @i P / R(i,i) priority 1; //Reflexivity
    R(i,j) R(j,k) / R(i,k) priority 2; //Transitivity
    @i P @i~P / priority 0; //Closure rule
    R(i,j) / [i=j] || ~([i=j]) priority 6; //Ancestor blocking rule
    ~([i=i])/ priority 0; //Closure rule for inequality
}
```

Outline

MANCHESTER 1824 The University of Manchester

Labelled Tableaux

```
sort labelledFormula, formula, label;
...
labelledFormula at = '@' label formula;
...
```

Example

```
syntax S4{
    sort tableauFormula, formula, world;
    formula false = 'false';
    formula negation = '~' formula;
    formula diamond = '<>' formula;
    tableauFormula at = '@' world formula;
    formula disjunction = formula '|' formula;
    tableauFormula equality = '[' world '=' world ']'; //Equality
    tableauFormula inequality = '~' '[' world '=' world ']'; //Equality
    tableauFormula relation = 'R' '(' world ',' world ')'; //Relation
    world f = 'f' '(' world ',' formula ')'; //Skolem function
}
```

Signed Tableaux

```
sort signedFormula, formula;
...
signedFormula trueValue = 'T' formula;
signedFormula falseValue = 'F' formula;
...
```

Example

```
syntax Int{
    sort formula, individual;
    formula false = 'false';
    formula trueValue = 'T' formula;
    formula falseValue = 'F' formula;
    formula at = '@' individual formula;
    formula negation = '~' formula;
    formula conjunction = formula '&' formula;
    formula disjunction = formula '|' formula;
    formula implication = formula '->' formula;
    individual successorImp = 'f' '(' individual ',' formula ',' formula ')';
    formula relation = 'R' '(' individual ',' individual ')';
    formula equality = '[' individual '=' individual ']';
}
```

Embedding Sorts

```
sort formula, proposition;
...
formula proposition = '#' proposition;
...
```

```
syntax IEL{
    sort formula, individual, prop, agent;
    formula true = 'true';
    formula false = 'false';
    formula singleton = '{' individual '}';
    formula atom = '#' prop;
    formula negation = '~' formula;
    formula diamondQ = '<q>' agent formula;
    formula diamondK = '<k>' agent formula;
    formula diamondX = '<x>' agent formula;
    formula at = '@' individual formula;
    formula query = '[?' formula ']' agent formula;
    formula resol = '[!]' agent formula;
    formula disjunction = formula '|' formula;
    formula equality = '[' individual '=' individual ']';
    individual fq = 'fq' '(' individual ',' agent ',' formula ')';
    individual fk = 'fk' '(' individual ',' agent ',' formula ')';
    individual fx = 'fx' '(' individual ',' agent ',' formula ')';
}
```

Eliminating Expressions

Example

```
options{
    ...
    equality.keywords={equality,equivalence}
}
syntax LTL{
    ...
    formula equivalence = formula '<->' formula;
    ...
    formula eventuality = 'E' '(' world ',' formula ')';
}
tableau LTL{
    ...
    @i (<>P) / @i (E(i,P)) priority 1;
    @i (E(j,P)) / @i P ((E(j,P)) <-> (<>P)) || @f(i) (E(j,P)) priority 7;
    ...
    @i E(j,P) / priority 8;//"Bad" loop check
}
```

Reducing Rule Branching Factor

Move negated conclusion to premises if it cannot be instantiated to become "non-atomic".

$$\frac{}{\neg R(i,j) \mid \neg R(j,k) \mid R(i,k)}$$

$$\frac{@_i \neg\Diamond p}{\neg R(i,j) \mid @_j \neg p}$$

Reducing Rule Branching Factor

Move negated conclusion to premises if it cannot be instantiated to become "non-atomic".

$$\frac{R(i,j), \quad R(j,k)}{R(i,k)}$$

$$\frac{@_i \neg \Diamond p}{\neg R(i,j) \mid @_j \neg p}$$

MANCHESTER 1824 The University of Manchester

Reducing Rule Branching Factor

Move negated conclusion to premises if it cannot be instantiated to become "non-atomic".

$$\frac{R(i,j),\;\; R(j,k)}{R(i,k)}$$

$$\frac{@_i \neg\Diamond p}{\neg R(i,j) \mid @_j \neg p}$$

MANCHESTER 1824 The University of Manchester

Reducing Rule Branching Factor

Move negated conclusion to premises if it cannot be instantiated to become "non-atomic".

$$\frac{R(i,j),\ \ R(j,k)}{R(i,k)}$$

$$\frac{@_i\neg\Diamond p,\ \ R(i,j)}{@_j\neg p}$$

Tabular Logics

```
specification Lukasiewicz3;
syntax Lukasiewicz{
    sort valuation, formula;
    valuation true = 'T' formula | unknown = 'U' formula | false = 'F' formula;
    formula true = 'true' | false = 'false';
    formula negation = '~' formula;
    formula conjunction = formula '&' formula;
    formula disjunction = formula '|' formula;
    formula implication = formula '->' formula;
}
tableau Lukasiewicz{
T P F P / priority 0 $; T P U P / priority 0 $;
U P F P / priority 0 $; U P F P / priority 0 $;
T ~P / F P priority 1$; U ~P / U P priority 1$; F ~P / T P priority 1$;
T (P & Q) / T P T Q priority 2$; F (P & Q) / F P $| F Q priority 1$;
    U (P & Q) / T P U Q $| U P T Q $| U P U Q priority 3$;
T (P | Q) / T P $| T Q priority 2$; F (P | Q) / F P F Q priority 1$;
    U (P | Q) / F P U Q $| U P F Q $| U P U Q priority 3$;
F (P -> Q) / T P F Q priority 1$; U (P -> Q) / U P F Q $| T P U Q priority 2$;
    T (P -> Q) / T Q $| F P $| U P U Q priority 3$;
T false / priority 0$; U false / priority 0$;
U true / priority 0$; F true / priority 0$;
}
```

Outline

MANCHESTER 1824 The University of Manchester

Options for Random Problem Generators

Copy **S4RandomExpressionGenerator.properties** from **output/S4/etc** and edit it.

```
world.f.frequency = 1

world.variable.frequency = 1
world.depth = 1
world.variables = i, j, k
world.variables.number = 3

world.generate = 0

world.top.connectives =

formula.false.frequency = 1
formula.negation.frequency = 1
formula.diamond.frequency = 1
formula.at.frequency = 1
formula.disjunction.frequency = 1
formula.worldEquality.frequency = 1
formula.relation.frequency = 1

formula.variable.frequency = 1
formula.depth = 10
formula.variables = p, q, r
formula.variables.number = 3

formula.generate = 1000

formula.top.connectives = at
```

Generating Random Problems and Benchmarking

- Generate problems

```
> java -cp S4.jar:mettel2-util.jar \
  S4.language.S4.util.S4RandomExpressionGenerator \
  -p S4RandomExpressionGenerator.properties
```

- Run a benchmark

```
> java -cp S4.jar:mettel2-util.jar:opencsv.jar \
  S4.language.S4.util.S4Benchmark \
  -d random_problems
```

MANCHESTER 1824 The University of Manchester

Outline

Summary

- METTEL^2 is easy-to-use and flexible tool
- METTEL^2 allows specification of various types of tableau calculi
- Case sudies:
  - Boolean logic
  - S4
  - IPC
  - $\mathcal{ALCO}$
  - $\mathcal{ALBO}^{\mathrm{id}}$
  - $\mathcal{SHOI}$
  - LTL and temporal logic with cardinality constraints
  - Tree-valued Łukasiewicz logic
  - K with global counting operators
  - Interrogative epistemic logics
  - etc

MANCHESTER 1824 The University of Manchester

Outlook

- Local aims:
  - Further optimisations of generated provers.
  - More options to allow the user to control the generation process.
  - Extending specification languages.
  - Increase level of abstraction of the tableau core.
- An *ultimate goal* is to enable automatic generation of provers from other definitions of logics. In particular:
  - Implement the tableau synthesis framework for synthesis of tableau calculi from semantics of logics.
  - Investigate a possibility to generate tableau provers from Hilbert axiomatisations.

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!

Thank You!