FCUBE: An Efficient Prover for Intuitionistic Propositional Logic

Mauro Ferrari[1] Camillo Fiorentini[2] Guido Fiorino[3]

[1]Dipartimento di Informatica e Comunicazione, Università degli Studi dell'Insubria,

[2]Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, Italy

[3]Dipartimento di Metodi Quantitativi per le Scienze Economiche Aziendali, Università degli Studi di Milano-Bicocca

LPAR
17th International Conference on Logic for Programming, Artificial Intelligence and Reasoning

Yogyakarta, Indonesia
October 10th - 15th, 2010

fCube

fCube

- is a Theorem prover for propositional Intuitionistic logic (**Int**),
- based on a tableau calculus,
- written in prolog,
- that returns a proof or a counter model.

fCube implements a family of techniques that reduce the search space acting on different aspects of the proof search:

- new rules, based on the replacement of a formula with a logical constant. **The rules can be applied to any complete tableau calculus for Int.**
- a strategy to minimize the handling of branching formulas (β-formulas);
- a strategy to reduce the backtracking.

There was very little work to optimize the implementation of fCube

fCube Tableau Calculus: an Overview

Based on the Tableau calculus of Avellone et al, TCS 2008
Signs **T** and **F**, as used by Smullyan and Fitting, to be understood as follows:

- **T**A means “A is proved to be true” or “A is known”;
- **F**A means “‘A is not proved to be true” or “A is not known”;

In terms of Kripke semantics:

- **T** means forcing;
- **F** means non-forcing:

Given a Kripke model $\underline{K} = \langle P, \leq, \rho, V \rangle$, $\alpha \in P$

- α *satisfies* **T**A *in* $\underline{K}$ iff $\alpha \Vdash A$;
- α *satisfies* **F**A *in* $\underline{K}$ iff $\alpha \nVdash A$.

- Match with sequent systems:
  T-formulas are on the left side of $\Rightarrow$;
  F-formulas are on the right of $\Rightarrow$.

FCUBE Tableau Calculus for **Int**

$$\frac{\Delta, \mathbf{T}(A \land B)}{\Delta, \mathbf{T}A, \mathbf{T}B}\,\mathbf{T}\land \qquad \frac{\Delta, \mathbf{F}(A \land B)}{\Delta, \mathbf{F}A \mid \Delta, \mathbf{F}B}\,\mathbf{F}\land \qquad \frac{\Delta, \mathbf{T}\neg(A \land B)}{\Delta_{\mathbf{T}}, \mathbf{T}\neg A \mid \Delta_{\mathbf{T}}, \mathbf{T}\neg B}\,\mathbf{T}\neg\land$$

$$\frac{\Delta, \mathbf{T}(A \lor B)}{\Delta, \mathbf{T}A \mid \Delta, \mathbf{T}B}\,\mathbf{T}\lor \qquad \frac{\Delta, \mathbf{F}(A \lor B)}{\Delta, \mathbf{F}A, \mathbf{F}B}\,\mathbf{F}\lor \qquad \frac{\Delta, \mathbf{T}\neg(A \lor B)}{\Delta, \mathbf{T}\neg A, \mathbf{T}\neg B}\,\mathbf{T}\neg\lor$$

$$\frac{\Delta, \mathbf{T}A, \mathbf{T}(A \to B)}{\Delta, \mathbf{T}A, \mathbf{T}B}\,MP \qquad \frac{\Delta_{\mathbf{T}}, \mathbf{T}(A \to B)}{\Delta_{\mathbf{T}}, \mathbf{T}\neg A \mid \Delta_{\mathbf{T}}, \mathbf{T}B}\,\mathbf{T}\!\to\!\text{-special}$$

$$\frac{\Delta, \mathbf{F}(A \to B)}{\Delta_{\mathbf{T}}, \mathbf{T}A, \mathbf{F}B}\,\mathbf{F}\to \qquad \frac{\Delta, \mathbf{T}\neg(A \to B)}{\Delta_{\mathbf{T}}, \mathbf{T}A, \mathbf{T}\neg B}\,\mathbf{T}\neg\to \qquad \frac{\Delta, \mathbf{F}\neg A}{\Delta_{\mathbf{T}}, \mathbf{T}A}\,\mathbf{F}\neg \qquad \frac{\Delta, \mathbf{T}\neg\neg A}{\Delta_{\mathbf{T}}, \mathbf{T}A}\,\mathbf{T}\neg\neg$$

$$\frac{\Delta, \mathbf{T}((A \land B) \to C)}{\Delta, \mathbf{T}(A \to (B \to C))}\,\mathbf{T}\!\to\!\land \qquad \frac{\Delta, \mathbf{T}(\neg A \to B)}{\Delta_{\mathbf{T}}, \mathbf{T}A \mid \Delta, \mathbf{T}B}\,\mathbf{T}\!\to\!\neg \qquad \frac{\Delta, \mathbf{T}((A \lor B) \to C)}{\Delta, \mathbf{T}(A \to C), \mathbf{T}(B \to C)}\,\mathbf{T}\!\to\!\lor$$

$$\frac{\Delta, \mathbf{T}((A \to B) \to C)}{\Delta_{\mathbf{T}}, \mathbf{T}A, \mathbf{F}p, \mathbf{T}(p \to C), \mathbf{T}(B \to p) \mid \Delta, \mathbf{T}C}\,\mathbf{T}\!\to\to \quad \text{with } p \text{ a new atom}$$

where $\Delta_{\mathbf{T}} = \{\mathbf{T}A \mid \mathbf{T}A \in \Delta\}$

Introduction to fCube Optimization Rules

- The rules of previous slide are necessary to decide **Int**;
- We are ging to present rules useful to reduce the search space, but not necessary to decide **Int**,
- The fCube optimization machinery relies on the replacement of an occurrence of a formula with a logical constant;
- fCube gives to the replacement rules higher priority than the rules of the logical calculus.

The Replacement Rules in Our Setting

FCUBE replaces a formula with a logical constant:

- Replacement
  If a formula A is "*proved to be true*", then replace every occurrence of A with $\top$

  $$A \vee B \quad \rightsquigarrow_{A \text{ is true}} \quad \top \vee B$$

  Similarly, if A is "*proved to be false*", then replace A with $\bot$
- First benefit
  the size of the formula is reduced;
- Second benefit
  using the truth tables some connectives are erased:

  $$\top \vee B \quad \rightsquigarrow \quad \top$$

To sum up, we have rewritten $A \vee B$ as $\top$

Replacement rules in tableau/sequent systems

- Firstly introduced in modal and classical calculi:

  F. Massacci. Simplification: A general constraint propagation technique for propositional and modal tableaux. TABLEAUX'98, LNCS, *vol. 1397, pp. 217–231.*

- In intuitionistic calculi:

  A. Avellone, G. Fiorino, and U. Moscato. Optimization techniques for propositional intuitionistic logic and their implementation. TCS, 409(1):41–58, 2008.

$$\frac{\mathbf{T}A,\ \Delta}{\mathbf{T}A,\ \Delta[\top/A]}\ \mathit{Replace{-}}\mathbf{T}$$

$\Delta[\top/A]$:
replace all the occurrences of A in Δ with $\top$

$$\frac{\mathbf{T}\neg A,\ \Delta}{\mathbf{T}\neg A,\ \Delta[\bot/A]}\ \mathit{Replace{-}}\mathbf{T}\neg$$

$\Delta[\bot/A]$:
replace all the occurrences of A in Δ with $\bot$

Rules implemented in the propositional theorem prover PITP.

Simplifying proofs

$$\frac{\mathbf{F}(A \to (A \vee B) \wedge (A \vee C))}{\mathbf{T}A,\ \mathbf{F}((A \vee B) \wedge (A \vee C))}\,\mathbf{F}{\to} \text{ by instaniating rule } \frac{S, \mathbf{F}(A \to B)}{S_{\mathbf{T}}, \mathbf{T}A, \mathbf{F}B}\,\mathbf{F}{\to}$$

We could continue the derivation by using rule $\mathbf{F}\wedge$. In alternative:

Step 1: Replacement

The formula $\mathbf{T}A$ can be understood as "*A is proved to be true*".
We can keep $\mathbf{T}A$ and replace all the other A with $\top$

$$\mathbf{T}A,\ \mathbf{F}((A \vee B) \wedge (A \vee C)) \quad \rightsquigarrow \quad \mathbf{T}A,\ \mathbf{F}((\top \vee B) \wedge (\top \vee C))$$

As a result of the replacement, A does not occur as proper subformula

Step 2: Boolean simplifications

$$\top \vee B \equiv \top \qquad \top \vee C \equiv \top \qquad \top \wedge \top \equiv \top$$

$$\mathbf{T}A,\ \mathbf{F}((\top \vee B) \wedge (\top \vee C)) \quad \rightsquigarrow \quad \underset{\text{inconsistent}}{\mathbf{T}A,\ \mathbf{F}\top}$$

What do we gain?

We have reduced the search space:

- In the first derivation, the rule $\mathbf{F}\wedge$ applied to

$$\mathbf{T}A,\ \mathbf{F}((A \vee B) \wedge (A \vee C))$$

  gives rise to two branches:

$$\mathbf{T}A,\ \mathbf{F}(A \vee B) \quad | \quad \mathbf{T}A,\ \mathbf{F}(A \vee C)$$

- In the second proof, by the application of simplification rules, $\mathbf{F}((A \vee B) \wedge (A \vee C))$ is rewritten as $\mathbf{F}\top$, and the branch point is eliminated.
- The replacement rules are invertible $\rightsquigarrow$ they are applicable at any step of the deduction without requiring backtracking. fCube gives to the replacement rules highest priority.

Further Optimizations Based on the Replacement Rules

We are going to introduce optimization rules that

1. are based on the replacement rules and
2. are invertible.

The theoretical results are presented in

Ferrari, Fiorentini, Fiorino.
Towards the use of simplification rules in intuitionistic tableaux.
In Marco Gavanelli and Fabrizio Riguzzi, editors, CILC09: 24-esimo Convegno Italiano di Logica Computazionale. *GULP, June 2009.*

The sign property

Given a signed formula $\mathcal{S}A$, ($\mathcal{S} \in \{\mathbf{T}, \mathbf{F}\}$), it is possible to determine the sign of the occurrences of the subformulas of A in the deductions (see, e.g., Kleene 52).

For example, given

$$\mathbf{F}((p \to q) \to (p \vee r))$$

in every possible deduction

- $(p \to q)$ and q occur with sign $\mathbf{T}$;
- $(p \vee r)$, r, p and p occur with sign $\mathbf{F}$.

Roughly speaking the sign of a formula determine the truth value necessary to fulfill it: $\mathbf{T}$ corresponds to $\top$, $\mathbf{F}$ corresponds to $\perp$.

Idea: the satisfiability of the above formula does not change if occurrence of p is replaced with $\perp$;

In the following our interest will be on the sign of propositional variables.

Positive occurrence

The following formal definition intends to express the fact that all the occurrences of a propositional variable p in a signed formula H have sign $\mathbf{T}$ or, equivalently, that in every possible tableau derivation for H the formula $\mathbf{F}p$ does not occur. Formally, the following defines when p positively occurs in a signed formula:

- $p \preceq^{+} \mathbf{T}p$
- $p \preceq^{+} \mathcal{S}\top$ and $p \preceq^{+} \mathcal{S}\bot \qquad \mathcal{S} \in \{\, \mathbf{T}, \mathbf{F} \,\}$
- $p \preceq^{+} \mathcal{S}q$, where q is any propositional variable such that $q \neq p$
- $p \preceq^{+} \mathcal{S}(A \odot B)$ iff $p \preceq^{+} \mathcal{S}A$ and $p \preceq^{+} \mathcal{S}B \qquad \odot \in \{\wedge, \vee\}$
- $p \preceq^{+} \mathbf{F}(A \to B)$ iff $p \preceq^{+} \mathbf{T}A$ and $p \preceq^{+} \mathbf{F}B$
- $p \preceq^{+} \mathbf{T}(A \to B)$ iff $p \preceq^{+} \mathbf{F}A$ and $p \preceq^{+} \mathbf{T}B$
- $p \preceq^{+} \mathbf{F}\neg A$ iff $p \preceq^{+} \mathbf{T}A$
- $p \preceq^{+} \mathbf{T}\neg A$ iff $p \preceq^{+} \mathbf{F}A$.

Note the parallelism with the rules for the formulas.

Erasing variables with constant sign

If all the occurrences of p in H have sign $\mathbf{T}$, then it is correct to replace p with $\top$ since in every deduction of H if p occurs, it occurs as the signed formula $\mathbf{T}p$.

Formally we can prove that

- If $p \preceq^+ \Delta$ (i.e., for every $H \in \Delta$, $p \preceq^+ H$), then:

$$\Delta \text{ is realizable} \quad \Longleftrightarrow \quad \mathbf{T}p, \Delta \text{ is realizable}$$

- As a consequence, the rule

$$\frac{\Delta}{Tp,\ \Delta}\ \mathbf{T}\text{-perm if } p \preceq^+ \Delta$$

  is invertible

- By scanning Δ once it is possible to find out the propositional variables having constant sign in Δ.

The Replacement Rules and Sign Permanence

The strategy of FCUBE is to instantiate the conclusion of the rule

$$\frac{\Delta}{Tp,\ \Delta}\ \mathbf{T}\text{-perm if } p \preceq^{+} \Delta$$

with the premise of the rule

$$\frac{\mathbf{T}A,\ \Delta}{\mathbf{T}A,\ \Delta[\top/A]}\,\text{Replace-}\mathbf{T}$$

In practice, in the FCUBE strategy every application of $\mathbf{T}$-perm is followed by an application of Replace-$\mathbf{T}$

An example from ILTP library

$$SYJ211 + 1.001 = \begin{array}{l} [(p0 \to p1)\& \\ ((\neg\neg p2 \to p3) \to p4)\& \\ ((\neg\neg p3 \to p4) \to p0)] \ \to p1 \end{array}$$

is a formula of ILTP library. All the occurrences of p2 in

$$\mathbf{F}(SYJ211 + 1.001) = \begin{array}{l} \mathbf{F}(((p0 \to p1)\& \\ \quad ((\neg\neg p2 \to p3) \to p4)\& \\ \quad ((\neg\neg p3 \to p4) \to p0)) \quad \to p1) \end{array}$$

are positive. After an application of $\mathbf{T}\text{-perm}$ and $\text{Replace-}\mathbf{T}$ we get

$$\mathbf{T}p2, \mathbf{F}(((p0 \to p1)\&((\neg\neg\top \to p3) \to p4)\&((\neg\neg p3 \to p4) \to p0)) \to p1)$$

By using the truth table of $\neg$ and $\to$ we get

$$\mathbf{F}(((p0 \to p1)\&(p3 \to p4)\&((\neg\neg p3 \to p4) \to p0)) \to p1)$$

that saves some backtracking steps. To decide SYJ211+1.010, a prover equipped with $\mathbf{T}\text{-perm}$ takes 0.01 secs, whereas without it takes more than 85secs.

Negative occurrence

If the all the occurrences of p in a signed formula H have sign **F**, then it is correct to replace p with $\bot$.

The formal definition of "have sign **F**" is given by $p \preceq^{-} H$, whose definition is analogous to $p \preceq^{+} H$.

We prove that:

- If all the occurrences of p in a set Δ have sign **F**, then:

$$\Delta \text{ is satisfiable} \quad \Longleftrightarrow \quad \mathbf{T}\neg p, \Delta \text{ is satisfiable}$$

As a consequence, the rule

$$\frac{\Delta}{T\neg p,\ \Delta}\ \mathbf{T}\neg\text{-perm} \text{ provided } p \preceq^{-} \Delta$$

is invertible.

Remark: $\mathbf{T}\neg p$ is a stronger condition than $\mathbf{F}p$
(in Kripke semantics: *local* falsity vs. *global* falsity)

Local Formulas and Branching Reduction

Rules $\mathrm{Replace}$-**T** and $\mathrm{Replace}$-**T**$\neg$ have highest priority, this implies that in the conclusions **T**A, $\Delta[\top/A]$ and **T**$\neg A$, $\Delta[\bot/A]$ there is exactly one occurrence of A.

Thus if a propositional variable p occurs in a non-atomic formula of Δ, then **T**$p \notin \Delta$ holds.

This remark is useful because fCube can avoid to apply rules to the family of **local formulas**.

The **local formulas** are the family of **F**-signed formulas defined as follows:

$$L ::= p \mid L \vee L \mid L \wedge A \mid A \wedge L \quad \text{where } p \in \mathcal{PV} \text{ and } A \text{ is any formula}$$

Local Formulas and fCube Strategy

Local formulas have the following property:

Theorem

Let $\underline{K} = \langle P, \leq, \rho, V \rangle$ be a Kripke model, $\alpha \in P$ and $\mathbf{F}L$ a local formula. If $\alpha \nVdash p$ for every p occurring in L, then $\alpha \nVdash L$.

Consequence of the theorem and of the highest priority given to the replacement rules:

To get the completeness is not necessary
to decompose the local formulas

Advantage: fCube does not handle local formulas of the kind $\mathbf{F}(A \wedge B)$, thus avoiding to insert into the proof the branches related to

$$\frac{\Delta, \mathbf{F}(A \wedge B)}{\Delta, \mathbf{F}A \mid \Delta, \mathbf{F}B}\,\mathbf{F}\wedge$$

Priority of the Rules in fCube

Invertible rules have higher priority than non-invertible rules;

Both invertible and non-invertible rules with one conclusion have higher priority than rules with two conclusions;

The replacement and permanence rules are all invertible with one conclusion.

To decide the application of the permanence rules a computation in linear time is required, thus fCube gives to the invertible rules with one conclusion this priority:

1. replacement rules;
2. rules of the logical calculus;
3. permanence rules.

Backtracking Reduction

The logical calculus contains eight non-invertible rules. Backtracking is unavoidable when the following rules are used

$$\frac{\Delta, \mathbf{F}(A \to B)}{\mathbf{\Delta_T}, \mathbf{TA}, \mathbf{FB}}\,\mathsf{F}{\to} \quad \frac{\Delta, \mathbf{T}((A \to B) \to C)}{\mathbf{\Delta_T}, \mathbf{TA}, \mathbf{Fp}, \mathbf{T(p \to C)}, \mathbf{T(B \to p)} \mid \Delta, \mathbf{T}C}\,\mathsf{T}{\to}{\to} \quad \frac{\Delta, \mathbf{F}\neg A}{\mathbf{\Delta_T}, \mathbf{TA}}\,\mathsf{F}\neg \quad \frac{\Delta, \mathbf{T}(\neg A \to B)}{\mathbf{\Delta_T}, \mathbf{TA} \mid \Delta, \mathbf{T}B}\,\mathsf{T}{\to}\neg$$

Given a set Δ, if from an instantiation of the primise of a rule above is not possible to get a proof of the conclusion, then there exists a Kripke model $\underline{K}'$ of the conclusion, **but $\underline{K}'$ is not necessarily a model of Δ**.

To be complete FCUBE has to try to instantiate in all possible ways the rules above with Δ.

Condition to stop the iteration

If $\underline{K}'$ satisfies Δ, then the backtracking can be stopped without affect the completeness. To check if $\underline{K}'$ satisfies Δ is sufficient to verify that $\underline{K}'$ satisfies the F-formulas in Δ.

propositional Intuitionistic Provers and ILTP Library

The Intuitionistic Logic Theorem Proving (ILTP) library provides a platform for testing and benchmarking automated theorem proving systems for first-order and propositional intuitionistic logic;

A list of theorem provers for propositional intuitionistic logic: **Imogen**, PITP, STRIP, LJT, ft-C and FCUBE;

Imogen, a theorem prover for intuitionistic propositional logic using the focused inverse method:

> *S. McLaughlin and F. Pfenning.*
> *Imogen: Focusing the Polarized Inverse Method for Intuitionistic Propositional Logic.*

Presented at Lpar 2008, it is the fastes theorem prover of ILTP library and decides 261 formulas.

We compare FCUBE with **Imogen**.

Comparing Imogen and FCUBE

Formula	Imogen	FCUBE
SYJ201+1.018	11.32	14.46
SYJ201+1.019	16.28	17.84
SYJ201+1.020	17.00	22.34
SYJ202+1.006	timeout	6.18
SYJ202+1.007	timeout	52.98
SYJ202+1.008	timeout	529.36
SYJ206+1.018	2.26	0.00
SYJ206+1.019	2.12	0.00
SYJ206+1.020	2.14	0.00
SYJ207+1.018	77.02	4.72
SYJ207+1.019	104.38	6.08
SYJ207+1.020	143.32	7.44
SYJ208+1.015	timeout	174.22
SYJ208+1.016	timeout	286.56
SYJ208+1.017	timeout	472.07
SYJ209+1.018	0.20	0.08
SYJ209+1.019	0.024	0.09
SYJ209+1.020	0.028	0.09
Nishimura.011	8.2	0.02
Nishimura.012	132	0.04
Nishimura.013	timeout	0.07

Imogen outperforms FCUBE on SYJ201 family formulas of ILTP Library;

reasons for these performances: FCUBE lacks of advanced data structures to store once multiple occurrences of the same formula. Note that:

1. $\wedge_{j=0}^{40} p_j$ occurs 42-times in SYJ201+1.020;
2. **FCUBE proof of SYJ201+1.020 contains 246 nodes**;

Experiment: SYJ201+1.020 rewritten as an equivalent formula where $\wedge_{j=0}^{40} p_j$ occurs once and the other occurrences are replaced by a propositional variable. Results:

1. **FCUBE decides the new formula in 2.23 secs** (Imogen in 33secs);
2. **FCUBE proof of the new version of SYJ201+1.020 contains 250 nodes**.

Comparing the Optimizations

Formula	fCube	Basic	+BackT	+Branch
SYJ201+1.018	14.46	timeout	timeout	16.16
SYJ201+1.019	17.84	timeout	timeout	20.72
SYJ201+1.020	22.34	timeout	timeout	26.00
SYJ202+1.006	6.18	timeout	timeout	7.08
SYJ202+1.007	52.98	timeout	timeout	61.18
SYJ202+1.008	529	timeout	timeout	570
SYJ206+1.018	0.00	0.00	0.00	0.00
SYJ206+1.019	0.00	0.00	0.00	0.01
SYJ206+1.020	0.00	0.01	0.01	0.01
SYJ207+1.018	4.72	timeout	5.53	timeout
SYJ207+1.019	6.08	timeout	7.15	timeout
SYJ207+1.020	7.44	timeout	8.94	timeout
SYJ208+1.015	174	220	209	187
SYJ208+1.016	286	351	349	312
SYJ208+1.017	472	570	569	541
SYJ209+1.018	0.08	timeout	0.07	timeout
SYJ209+1.019	0.09	timeout	0.09	timeout
SYJ209+1.020	0.09	timeout	0.10	timeout
Nishimura.011	0.02	0.02	0.02	0.02
Nishimura.012	0.04	0.03	0.04	0.04
Nishimura.013	0.07	0.06	0.07	0.07

Basic: logical calculus + replacement and permanence rules;

+BackT: Basic + Backtracking Reduction

+Branch: Basic + Local Formulas;

fCube: Basic + Backtracking Reduction + Local Formulas;

Conclusions

FCUBE is a prolog theorem prover for propositional intuitionistic logic;

FCUBE provides a proof or a countermodel;

the main effort has been devoted to develop optimization techniques to reduce the search space;

despite the lack of advanced data structures FCUBE outperforms Imogen, the fastest theorem prover for propositional intuitionistic logic;

we have extended the results related to permanence rules and we are working to apply the ideas presented in this talk to the first-order case.

Making the replacement more general

The rules $\mathbf{T}$-perm and $\mathbf{T}\neg$-perm can be made more general. Let us start from a simple case base:

$$\mathbf{F}p, \mathbf{F}q, \mathbf{F}((p \vee r) \rightarrow q), \mathbf{T}(r \rightarrow q)$$

In this formula every propositional variable occurs both with sign $\mathbf{T}$ and with sign $\mathbf{F}$.

We recall the rule to handle $\mathbf{F} \rightarrow$-formulas ($R \rightarrow$-formulas)

$$\frac{S, \mathbf{F}(A \rightarrow B)}{S_{\mathbf{T}}, \mathbf{T}A, \mathbf{F}B}\,\mathbf{F}{\rightarrow} \qquad\qquad \frac{\Delta, A \Rightarrow B}{\Delta \Rightarrow \Theta, (A \rightarrow B)}\,R{\rightarrow}$$

When the premise of $\mathbf{F} \rightarrow$ is instantiated by setting

$$\mathbf{F}(A \rightarrow B) = \mathbf{F}((p \vee r) \rightarrow q) \text{ and } S = \{\mathbf{F}p, \mathbf{F}q, \mathbf{T}(r \rightarrow q)\}$$

the conclusion is

$$\mathbf{T}(p \vee r), \mathbf{F}q, \mathbf{T}(r \rightarrow q)$$

where now p can be replaced with $\top$.

Making the replacement more general

What we notice is that given

$$\mathbf{F}p, \mathbf{F}q, \mathbf{F}((p \vee r) \to q), \mathbf{T}(r \to q)$$

inside

$$\mathbf{F}((p \vee r) \to q)$$

the variable p can be replaced by $\top$ because we can forecast that for every deduction, when $\mathbf{F} \to$ is applied, in the conclusion p will occur only with sign $\mathbf{T}$.

As a matter of fact:

1. $p \preceq^{+} F((p \vee r) \to q)$;
2. $p \preceq^{+} S_{\mathbf{T}} = \{\mathbf{T}(r \to q)\}$.

Thus we get the sequence of sets:

$$\mathbf{F}p, \mathbf{F}q, \mathbf{F}((\top \vee r) \to q), \mathbf{T}(r \to q)$$

$$\mathbf{F}p, \mathbf{F}q, \mathbf{F}(\top \to q), \mathbf{T}(r \to q)$$

$$\mathbf{F}p, \mathbf{F}q, \mathbf{F}q, \mathbf{T}(r \to q)$$

An example from ILTP library: unprovable de Bruijn's formulas

To prove the de Bruijn formula SYJ207+1.002, we start from

$$\mathbf{F}(\; ((p0 \leftrightarrow p1) \to A) \wedge ((p1 \leftrightarrow p2) \to A) \;\wedge$$
$$((p2 \leftrightarrow p3) \to A) \wedge ((p3 \leftrightarrow p0) \to A)) \quad)$$
$$\to (p4 \vee A \vee (p4 \to \bot)$$

where $A = (p0 \wedge (p1 \wedge (p2 \wedge p3)))$.

After few steps we come up with

$$\mathbf{T}((p0 \leftrightarrow p1) \to A),\ \mathbf{T}((p1 \leftrightarrow p2) \to A),\ \mathbf{T}((p2 \leftrightarrow p3) \to A),$$
$$\mathbf{T}((p3 \leftrightarrow p0) \to A),\ \mathbf{F}p4,\ \mathbf{F}A,\ \mathbf{F}(p4 \to \bot)$$

By applying $\mathbf{F} \to$ to $\mathbf{F}(p4 \to \bot)$ we get a conclusion where $p4$ occurs once. Inside $\mathbf{F}(p4 \to \bot)$ it is correct to replace $p4$ with $\top$.

As a result $\mathbf{F}(p4 \to \bot)$ disappear and this saves many backtracking steps.

FCUBE implements this rule and takes 7.44secs to decide SYJ207+1.020. FCUBE lacking of this version of permanence rule takes 670secs to decide SYJ207+1.020.

The rule $\mathbf{T}$-cperm

The rule we have discussed is the following:

$$\frac{\Delta, \mathbf{F}(A \to B)}{\Delta, \mathbf{F}(A[\top/p] \to B[\top/p])}\ \mathbf{T}\text{-cperm}, \text{ provided all the occurrences of } p \text{ in } \mathbf{F}(A \to B) \cup \Delta_{\mathbf{T}} \text{ are positive}$$