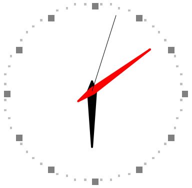


# Consequence finding with SOLAR-C

**NABESHIMA Hidetomo**

**University of Yamanashi**

17th Sep 2007
Aix-en-Provence

# Contents

- Introduction of **SOLAR**
- Demonstration
- Two kinds of SOLARs: **SOLAR-J** and **SOLAR-C**
- Progress of development
- Experimental results
- Future work and Summary

# Introduction

**SOLAR** (SOL for Advanced Reasoning)

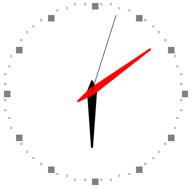
Efficient implementation of *consequence finding* procedure ***SOL***

- What is *consequence finding?*
  - find *important consequences* from an axiom set
  - generalization of refutation finding or theorem proving

⬇

The set of theorems is generally *infinite*, even if they are restricted to be minimal wrt subsumption.

# Finding Interesting Consequences



[Inoue, 90;91;92] reformulated the problem as follows:

How to find only ***interesting*** consequences?

**Solutions**

***Production field*** and ***characteristic clauses***

plus

**SOL procedure** (Skipping Ordered Linear resolution),

(a model-elimination-like calculus with Skip operation)

# Production Field

- **Production field**: $\boldsymbol{P} = \langle \boldsymbol{L}, \mathit{Cond} \rangle$
  - $\boldsymbol{L}$ : the set of literals to be collected
  - $\mathit{Cond}$ : the condition to be satisfied (e.g. length)
- $Th_{\boldsymbol{P}}(\Sigma)$ **:** the clauses entailed by $\Sigma$ which belong to $\boldsymbol{P}$.

☐ $\boldsymbol{P1} = \langle \{ANS\}^+, \text{ none} \rangle$ :

- $\{ANS\}^+$ is the set of positive literals with the predicate $ANS$.
- $Th_{\boldsymbol{P1}}(\Sigma)$ is the set of all positive clauses of the form $ANS(t_1) \lor \ldots \lor ANS(t_n)$ being derivable from $\Sigma$.

☐ $\boldsymbol{P2} = \langle \boldsymbol{L}^-, \text{ length is fewer than } k \rangle$:

- $\boldsymbol{L}^-$ is the set of negative literals.
- $Th_{\boldsymbol{P2}}(\Sigma)$ is the set of all negative clauses derivable from $\Sigma$ consisting of fewer than $k$ literals.

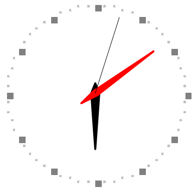
# Characteristic Clauses

- **Characteristic clause** of $\Sigma$ (wrt $\boldsymbol{P}$):
  - A clause $D$ is a characteristic clause if
    - $D$ belongs to $Th_{\boldsymbol{P}}(\Sigma)$, and
    - no other clause in $Th_{\boldsymbol{P}}(\Sigma)$ subsumes $D$.
  - The set of characteristic clauses $Carc(\Sigma, \boldsymbol{P}) = \mu\, Th_{\boldsymbol{P}}(\Sigma)$,

    where μ represents "subsumption-minimal".
- **New characteristic clause** of $C$ wrt $\Sigma$ (and $\boldsymbol{P}$) :
  - A char. clause of $\Sigma \land C$ which is not a char. clause of $\Sigma$.
  - $NewCarc(\Sigma, C, \boldsymbol{P}) = \boldsymbol{\mu}[\, Th_{\boldsymbol{P}}(\Sigma \land C) - Th\ (\Sigma)\,]$

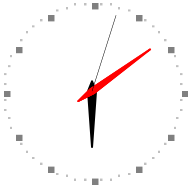
    $= Carc(\Sigma \land C, \boldsymbol{P}) - Carc(\Sigma, \boldsymbol{P})$.

# Most Special/General Production Field



- **Most special production field: *P* = <none>**
  - only the empty clause □ belongs to ***P*.**
  - **Refutation Finding (theorem proving)**
- **Most general production field: *P* = <all>**
  - *Carc*($\Sigma$,***P***) is the minimal theory of $\Sigma$ wrt subsumption.
  - In the propositional case, *Carc*($\Sigma$,***P***) is the set of **prime implicates** of $\Sigma$.

# Soundness & Completeness of SOL



- **SOL** can compute $NewCarc(\Sigma, C, \mathbf{P})$ and $Carc(\Sigma, \mathbf{P})$
- **Soundness**

  If a clause $S$ is derived by an SOL-deduction from $\Sigma + C$ and $P$, then $S$ belongs to $Th_{\mathbf{P}}(\Sigma \cup \{C\})$.
- **Completeness**

  If a clause $F$ does not belong to $Th_{\mathbf{P}}(\Sigma)$ but belongs to $Th_{\mathbf{P}}(\Sigma \cup \{C\})$, then there is an SOL deduction of a clause $S$ from $\Sigma + C$ and $P$ such that $S$ subsumes $F$.

# Demonstration

- Deduction
  - Dreadbury example
- Abduction
  - Graph completion

# Dreadbury Mansion

- Someone who lives in Dreadbury Mansion killed Aunt Agatha.
- Agatha, the butler, and Charles live in Dreadbury Mansion, and are the only people who live therein.
- A killer always hates his victim, and is never richer than his victim.
- Charles hates no one that Aunt Agatha hates.
- Agatha hates everyone except the butler.
- The butler hates everyone not richer than Aunt Agatha.
- The butler hates everyone Aunt Agatha hates.
- No one hates everyone.

*Who killed Agatha?*

# Dreadbury Example

- Someone who lives in Dreadbury Mansion killed Aunt Agatha.

$$killed(agatha, agatha) \vee killed(butler, agatha) \vee killed(charles, agatha)$$

- Agatha, the butler, and Charles live in Dreadbury Mansion.

$$lives(agatha) \wedge lives(butler) \wedge lives(charles)$$

- A killer always hates his victim, and is never richer than his victim.

$$killed(X,Y) \Rightarrow hates(X,Y) \wedge \neg richer(X,Y)$$

- Charles hates no one that Aunt Agatha hates.

$$hates(agatha, X) \Rightarrow \neg hates(charles, X)$$

# Dreadbury Example

- Agatha hates everyone except the butler.

$$hates(agatha, agatha) \wedge hates(agatha, charles)$$

- The butler hates everyone not richer than Aunt Agatha.

$$lives(X) \wedge \neg richer(X, agatha) \Rightarrow hates(butler, X)$$

- The butler hates everyone Aunt Agatha hates.

$$hates(agatha, X) \Rightarrow hates(butler, X)$$

- No one hates everyone.

$$\neg hates(X, agatha) \vee \neg hates(X, butler) \vee \neg hates(X, charles)$$

# Who killed Agatha?

For solving this problem, we define the production field as follows:

$$P = \{\, killed(\_, \_) \,\}$$

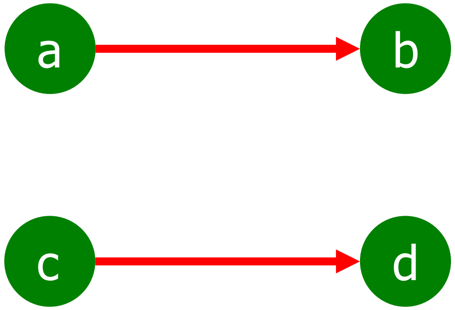
This means that we want to collect the consequences wrt “killed”.

## Demonstration

# Abduction Example
# Graph Completion

Let G be a graph (V,E)

- V={a, b, c, d}
- E={(a, b), (c, d)}



Background knowledge

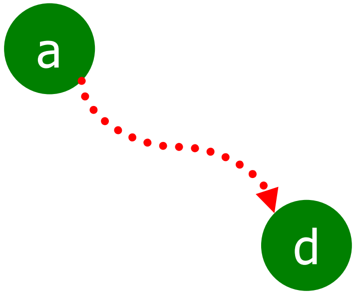
$$\Sigma \cup H \models O$$

$$\Downarrow$$

$$\Sigma \cup \neg O \models \neg H$$

$$\Downarrow$$

$$\Sigma \cup \neg path(a,d) \models \neg H$$



Observation

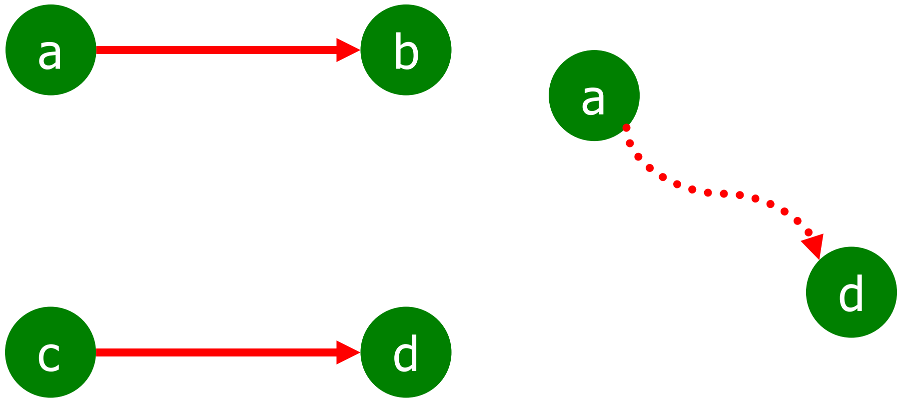
# Graph Completion

$$\begin{aligned}\Sigma = \{ & node(a), node(b), node(c), node(d),\\ & arc(a,b), arc(c,d),\\ & -node(X) \vee -node(Y) \vee -arc(X,Y) \vee path(X,Y),\\ & -node(X) \vee -node(Y) \vee -node(Z) \vee\\ & \qquad -arc(X,Y) \vee -path(Y,Z) \vee path(X,Z)\}\end{aligned}$$

Definition of "path"

$$\Sigma \cup \neg path(a,d) \models \neg H$$

$$P = \{-arc(\_,\_)\}$$



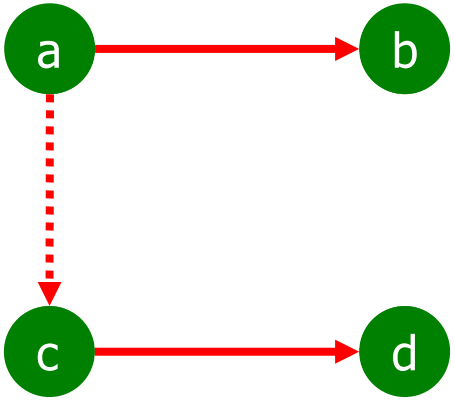
# Graph Completion Result

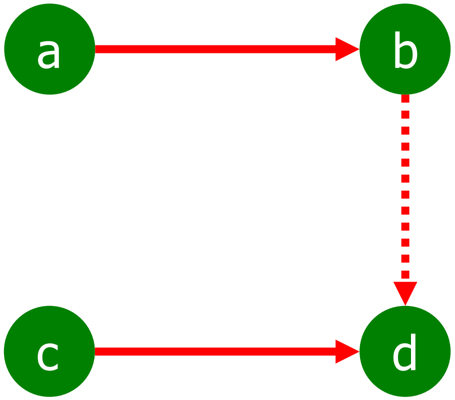
Demonstration

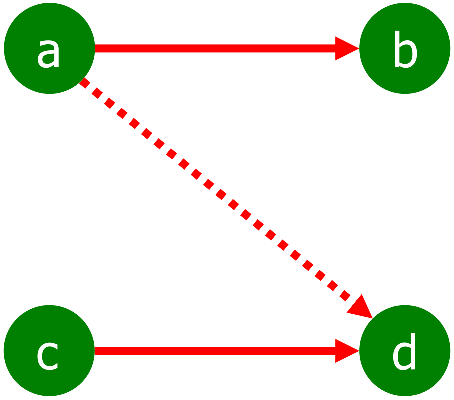
# Graph Completion Result

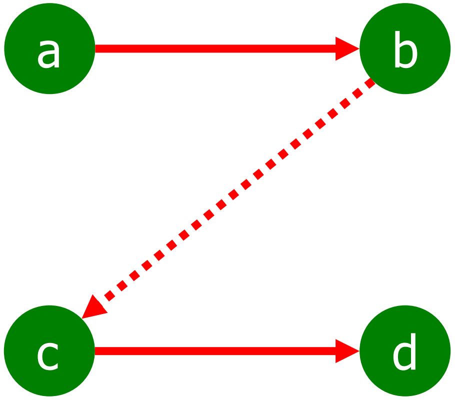
$$\Sigma \cup \neg path(a,d) \models \neg H$$

$$H = \{arc(a,c), arc(b,d), arc(a,d), arc(b,c)\}$$







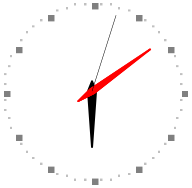


# Two Kinds of SOLARs

- Java version (**SOLAR-J**)
  - The first sophisticated implementation of consequence finding calculus in the world
    - Connection tableau format [Iwanuma et al., 2000]
    - Various pruning methods [Iwanuma et al., 2000, 2002]
  - As a theorem prover, the performance is comparable with Otter3.2

- C++ version (**SOLAR-C**)
  - Developing now
  - Purpose is to provide the ***more practical tool*** for various application areas

# Progress Status of SOLAR-C



- SOL Calculus
  - **Skip** ← Not implemented
  - **Extension** }
  - **Reduction** } Implemented

**Current SOLAR-C = SOLAR-J - Consequence finding**

**= Theorem prover**

# Progress Status of SOLAR-C

- SOL Calculus
  - **Skip** ← Not implemented
  - **Extension**
  - **Reduction**

  } Implemented

**SOLAR-J** totally 22000 lines

**Consequence finding part**
4000 lines

**Theorem proving part**
18000 lines

**Theorem proving part is the important basis of consequence finding system**

# Progress Status of SOLAR-C

**SOLAR-J** totally 22000 lines

**Consequence finding part**
4000 lines

**Theorem proving part**
18000 lines

Source code decreased to the half

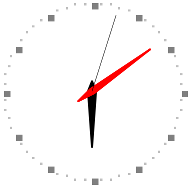
- unifying many routines which have similar functions
- redesigning core data structures

**SOLAR-C** currently 9000 lines

**Consequence finding part**
Not implemented

**Theorem proving part**
9000 lines

# Performance as Theorem Prover



- The TPTP Problem Library v2.5.0 consists of 6672 problems
- SOLAR can interpret CNF-style problems (**5181** problems) in TPTP library.

- Solved: # of proved problems
- Failed : over 1 CPU min or memory overflow
- Environment: Mac mini (Core Duo 1.66GHz, 2GB RAM)

| | 3260 problems containing equality | | | 1921 problems containing no equality | | |
|---|---|---|---|---|---|---|
| | Solved | Failed | Rate | Solved | Failed | Rate |
| **Otter 3.3f** | 888 | 2372 | 27.2% | 896 | 1025 | 46.6% |
| **SOLAR-J** | 591 | 2669 | 18.1% | 976 | 945 | 50.8% |
| **SOLAR-C** | 618 | 2642 | 19.0% | 1083 | 838 | 56.4% |

# Future Work

- Pruning methods for consequence finding [Iwanuma et al., 2000, 2002]
  - Regularity
  - Tautology-freeness
  - Complement-freeness
  - Skip-regularity
  - TCS-freeness

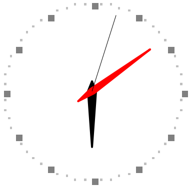
  :

Partially implemented in **SOLAR-J**

Full implementation in **SOLAR-C**

- Subgoal selection, literal ordering and clause ordering
  - In the current implementation, such orderings depend on the problem description (there is no *policy* or *heuristics*)
  - Consideration of ***weight*** concept of literal or clause (ex. the number of symbols in a literal or clause). Usually, light literals (clauses) are preferred.

# Future Work



- LRS (limited resource strategy) [Riazanov and Voronkov, 2003]
  - Important technique for practical use of saturation-based theorem provers

⬇

We would like to apply the LRS technique to SOLAR

## Basic idea

☐ If

- ✓ we know the processing rate of nodes per seconds, and
- ✓ we can estimate the number of nodes for closing a tableau

☐ then,

- ✓ we can ignore nodes which are estimated unclosable in the limited time, and try another possibility.

# Summary

- Introduction of SOL calculus and SOLAR-J
- Progress of SOLAR-C

Current implementation has many issues which should be solved

if solved...

*Performance will be improved! (greatly?)*

**Release schedule: Dec. 2007**

# Fin.