The Model Evolution Calculus with Equality

Peter Baumgartner

Max-Planck-Institute for Informatics

Saarbrücken, Germany

Cesare Tinelli

The University of Iowa

Iowa, USA

Background – Instance Based Methods

- **Model Evolution** is related to Instance Based Methods
  - Ordered Semantic Hyper Linking [Plaisted et al]
  - Primal Partial Instantiation [Hooker et al]
  - Disconnection Method [Billon], DCTP [Letz&Stenz]
  - Inst-Gen [Ganzinger&Korovin]
  - Successor of First-Order DPLL [B.]
- Principle: Reduce proof search in first-order (clausal) logic to propositional logic in an „intelligent“ way
- Different to Resolution, Model Elimination,...
  (Pro‘s and Con‘s)

Background – Model Evolution

- The best modern SAT solvers (satz, MiniSat, zChaff, Berkmin,...) are based on the Davis-Putnam-Logemann-Loveland procedure [DPLL 1960-1963]
- Can DPLL be lifted to the first-order level?
  How to combine
  - successful SAT techniques
    (unit propagation, backjumping, lemma learning,...)
  - successful first-order techniques?
    (unification, redundancy concepts, ...)?
- Realization in Model Evolution calculus / Darwin implementation
  - Basic approach developed (CADE-19)
  - Lemma learning on the way
  - **This work: built-in equality reasoning**

Contents

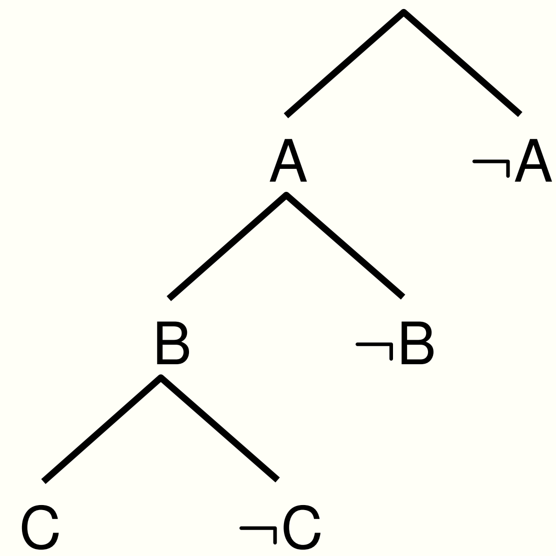
- DPLL as a starting point for the Model Evolution calculus
- Model Evolution calculus without equality
  - Model construction
  - Inference rules
- Equality reasoning
  - Equality reasoning in instance based methods
  - Inference rules
  - How it works (semantical considerations)

DPLL procedure

Input: Propositional clause set
Output: Model or „unsatisfiable"

Algorithm components:

- Propositional semantic tree
  enumerates interpretations
- Simplification
- Split
- Backtracking



Lifting to first-order logic?

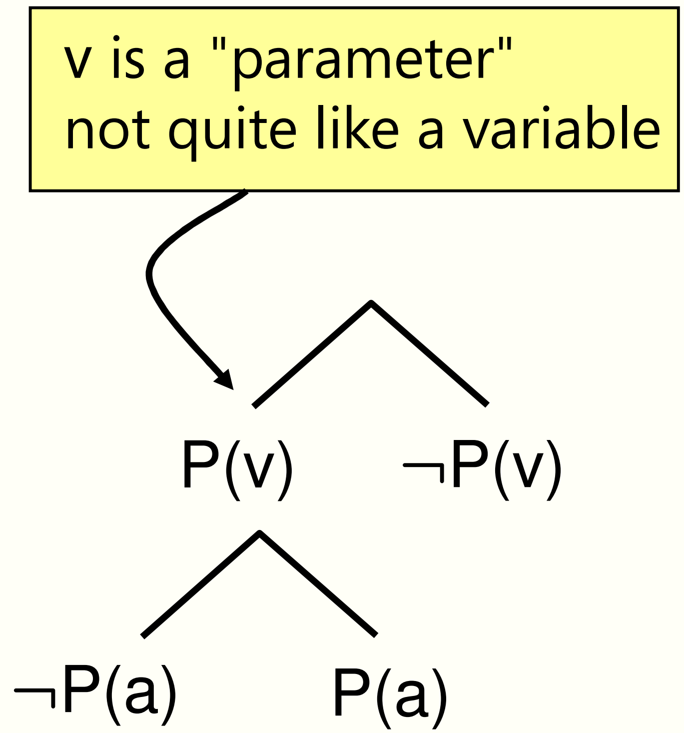
$$\{A, B\} \overset{?}{\models} \cancel{\neg A} \lor \cancel{\neg B} \lor C \lor D, \ldots$$

No, split on C: $\{A, B, C\} \models \cancel{\neg A \lor \neg B \lor C \lor D}, \ldots$

Model Evolution as First-Order DPLL

Lifing of semantic tree data structure and derivation rules to first-order

Input: First-order clause set
Output: Model or „unsatisfiable"
if termination



Algorithm components:

- First-order semantic tree enumerates interpretations
- Simplification
- Split
- Backtracking

$$\{P(v), \neg P(a)\} \overset{?}{\models} P(x) \vee Q(x)$$

Interpretation induced by a branch?

Interpretation Induced by a Branch

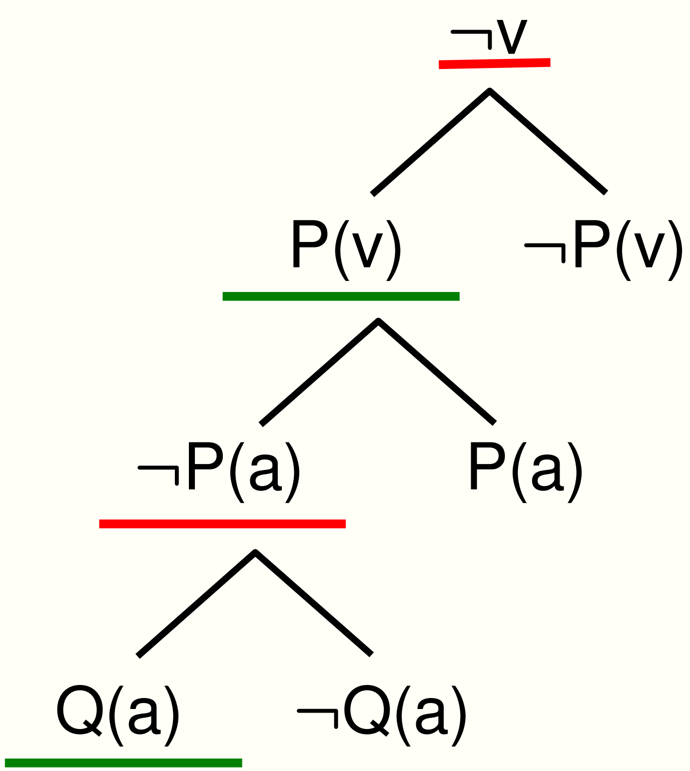
A branch literal specifies the truth value of its ground instances unless a more specific branch literal specifies the opposite truth value

Branch:
$\{\neg v,\ P(v),\ \neg P(a)\}$

Induced Interpretation for $\Sigma = \{a, b\}$

true: $P(b)$ $Q(a)$

false: $P(a)$ $\cancel{Q(a)}$ $Q(b)$



Calculus?

$$\{\neg v, P(v), \neg P(a)\} \overset{?}{\models} P(x) \lor Q(x)$$

No, because $\{\neg v, P(v), \neg P(a)\} \not\models P(a) \lor Q(a)$

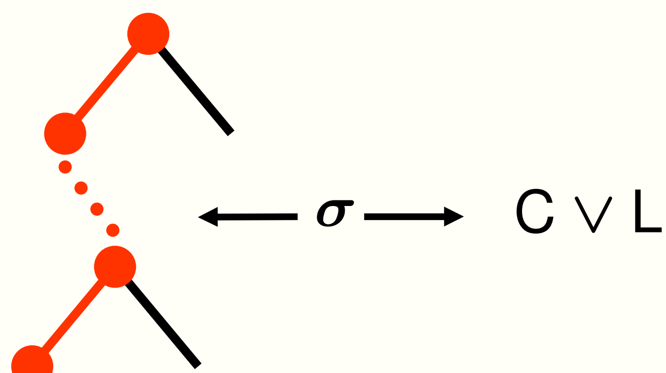
$\Rightarrow$ Split with $Q(a)$ to satisfy $P(a) \lor Q(a)$

Derivation Rules – Simplified (1)

$$\text{Split}\quad \frac{\Lambda \ \vdash \ \Phi,\ C \vee L}{\Lambda,\ L\boldsymbol{\sigma} \ \vdash \ \Phi,\ C \vee L \qquad \Lambda,\ \overline{L}\boldsymbol{\sigma} \ \vdash \ \Phi,\ C \vee L}$$

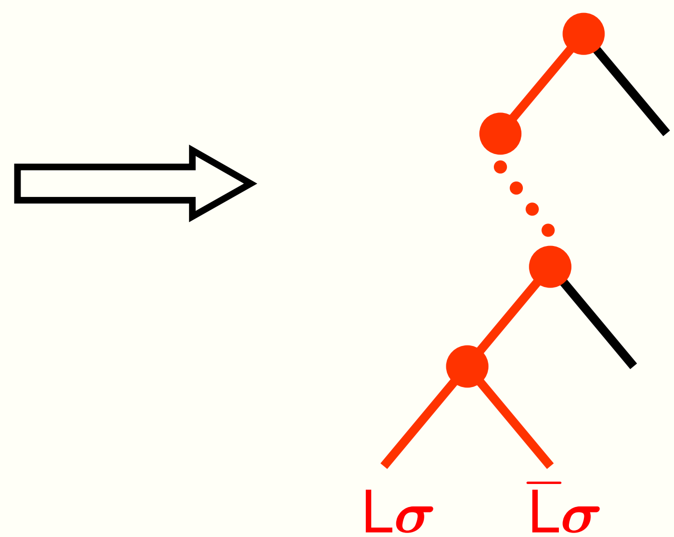
if

1. $\boldsymbol{\sigma}$ is a simultaneous mgu of $C \vee L$ against Λ,
2. neither $L\boldsymbol{\sigma}$ nor $\overline{L}\boldsymbol{\sigma}$ is contained in Λ, and
3. $L\boldsymbol{\sigma}$ contains no variables



Λ current context

$\Phi,\ C \vee L$ current clause set

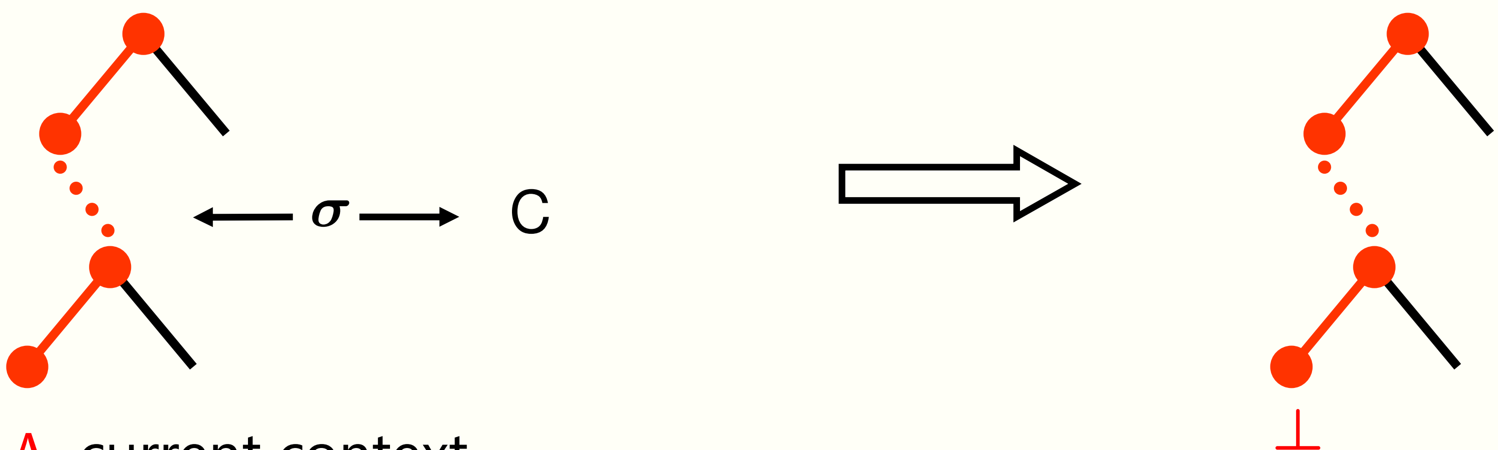


Derivation Rules – Simplified (2)

$$\text{Close}\quad \frac{\Lambda \;\vdash\; \Phi,\ C}{\Lambda \;\vdash\; \bot}$$

if

1. $\Phi \neq \emptyset$ or $C \neq \bot$
2. there is a simultaneous mgu $\boldsymbol{\sigma}$ of C against Λ such that Λ contains the complement of each literal of $C\boldsymbol{\sigma}$



Λ current context

Φ, C current clause set

Derivation Rules – Simplification Rules (1)

Propositional level:

$$\text{Subsume} \quad \frac{\Lambda,\ L \ \vdash \ \Phi,\ L \vee C}{\Lambda,\ L \ \vdash \ \Phi}$$

First-order level $\approx$ unit subsumption:

- L contains no parameters (variables OK)
- Matching instead of syntactic equality

Derivation Rules – Simplification Rules (2)

Propositional level:

Resolve
$$\frac{\Lambda,\ L \ \ \vdash \ \ \Phi,\ \overline{L} \lor C}{\Lambda,\ L \ \ \vdash \ \ \Phi,\ C}$$

First-order level $\approx$ restricted unit resolution

- L contains no parameters (variables OK)
- Unification instead of syntactic equality
- The unifier must not modify C

Derivation Rules – Simplification Rules (3)

$$\text{Compact} \quad \frac{\Lambda,\ K,\ L \ \vdash \ \Phi}{\Lambda,\ K \ \vdash \ \Phi}$$

if

1. K contains no parameters (variables OK)
2. $K\sigma = L$, for some substitution σ

Calculus

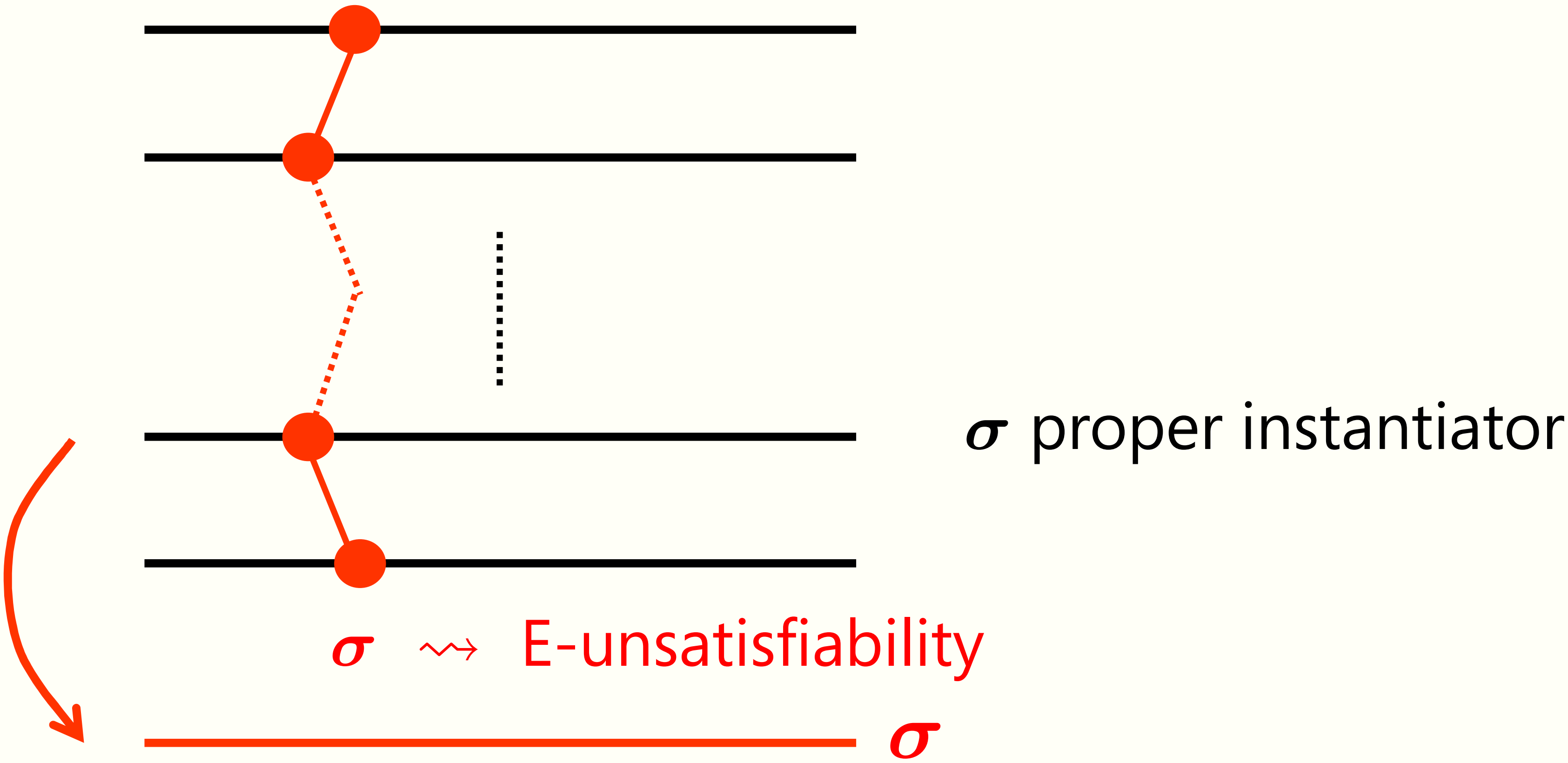
- Derivations are trees over sequents
- initial sequent $\neg v \vdash$ Input clause set
- Fairness
- Soundness and completeness

Contents

- DPLL as a starting point for the Model Evolution calculus
- Model Evolution calculus without equality
  - Model construction
  - Inference rules
- **Equality reasoning**
  - **Equality reasoning in instance based methods**
  - Inference rules
  - How it works (semantical considerations)

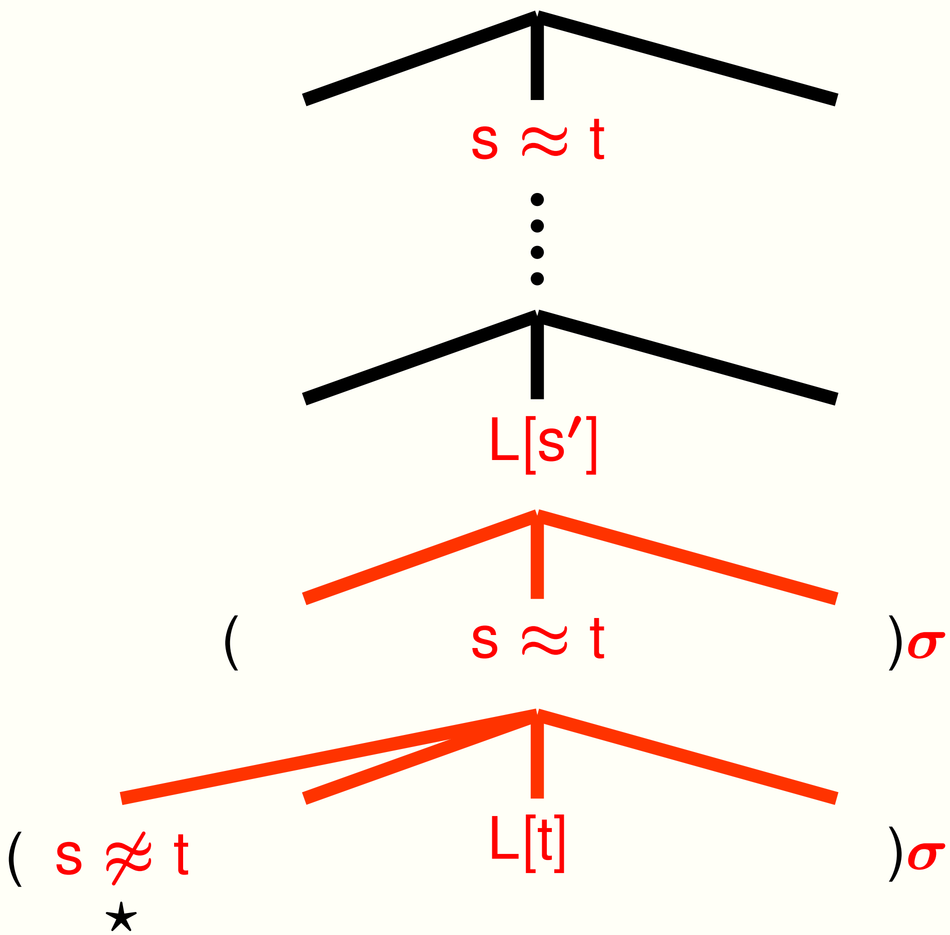
Equality Reasoning in Instance Based Methods

Inst-Gen [Ganzinger&Korovin CSL 2004]:



Equality Reasoning in Instance Based Methods

DCTP [Letz&Stenz Tableaux 02, Stenz 03]:



$\sigma = \mathrm{mgu}(s, s')$
$s\sigma \not\succeq t\sigma$

Our approach: related

Contents

- DPLL as a starting point for the Model Evolution calculus
- Model Evolution calculus without equality
  - Model construction
  - Inference rules
- **Equality reasoning**
  - Equality reasoning in instance based methods
  - **Inference rules**
  - How it works (semantical considerations)

Model Evolution Calculus with Equality - Overview

- **Split and Close**: same
- **Simplification rules**: general rule based on redundancy concept
- **Clauses**: with constraints now: $L_1 \vee \cdots \vee L_m \cdot l_1 \to r_1, \ldots, l_n \to r_n$
  Initially $C \cdot \emptyset$ for an input clause C

-**Equality reasoning**: Reflection and (Ordered) Paramodulation rules

Reflection

$$\frac{s \not\approx t \vee C \cdot \Gamma}{(C \cdot \Gamma)\sigma} \quad \text{if } \sigma = \mathrm{mgu}(s, t)$$

Paramodulation

Assumes reduction ordering $\succ$

$$\frac{l \approx r \quad L[t] \vee C \cdot \Gamma}{(L[r] \vee C \cdot \Gamma, l \to r)\sigma} \quad \text{if } \begin{cases} \sigma = \mathrm{mgu}(l, t) \\ t \text{ is not a variable} \\ l\sigma \not\preceq r\sigma \end{cases}$$

Embed these rules in calculus

Derivation Rules (1) - Reflection

$$\text{Ref}\ \frac{\Lambda \ \vdash \ \Phi,\ s \not\approx t \vee C \cdot \Gamma}{\Lambda \ \vdash \ \Phi,\ s \not\approx t \vee C \cdot \Gamma,\ (C \cdot \Gamma)\sigma}$$

if

1. σ is a mgu of s and t,
2. the new clause is not contained in $\Phi \cup \{s \not\approx t \vee C \cdot \Gamma\}$

Derivation Rules (2) - Paramodulation

$$\text{Para}\ \frac{\Lambda,\ l \approx r \quad \vdash \quad \Phi,\ L[t] \vee C \cdot \Gamma}{\Lambda,\ l \approx r \quad \vdash \quad \Phi,\ L[t] \vee C \cdot \Gamma,\ (L[r] \vee C \cdot \Gamma,\ l \rightarrow r)\boldsymbol{\sigma}}$$

if

1. $\boldsymbol{\sigma}$ is a mgu of l and t,
2. t is not a variable,
3. $l\boldsymbol{\sigma} \not\preceq r\boldsymbol{\sigma}$,
4. the new clause contains no parameters, and
5. the new clause is not contained in $\Phi \cup \{L \vee C \cdot \Gamma\}$

NB – This is not a resolution calculus:

- Paramodulation only from *unit* equations
- Clause part does not grow in length, and no paramodulation into constrained part
- (No paramodulation from context equations *into context literals)*

Derivation Rules (3) - Split

Split applies to

$$C \cdot l_1 \to r_1, \ldots, l_n \to r_n$$

only if C is a positive clause

Use conversion to ordinary clause

$$C \vee l_1 \not\approx r_1 \vee \cdots \vee l_n \not\approx r_n$$

and ordinary Split:

$$\text{Split} \quad \frac{\Lambda \;\vdash\; \Phi,\; C \vee L}{\Lambda,\; L\boldsymbol{\sigma} \;\vdash\; \Phi,\; C \vee L \qquad \Lambda,\; \overline{L}\boldsymbol{\sigma} \;\vdash\; \Phi,\; C \vee L}$$

Derivation Example

Initial clause encodes $\neg P(x, y) \lor Q(x) \lor R(y)$:

$$\neg v, \underline{P(u, u) \approx t} \quad \vdash \quad \underline{P(x, y) \not\approx t} \lor Q(x) \approx t \lor R(y) \approx t \cdot \emptyset$$

$$\Big\downarrow \ \text{Para}$$

$$\neg v, P(u, u) \approx t \quad \vdash \quad \begin{array}{r} P(x, y) \not\approx t \lor Q(x) \approx t \lor R(y) \approx t \cdot \emptyset, \\ \underline{t \not\approx t} \quad \lor Q(x) \approx t \lor R(x) \approx t \cdot P(x, x) \to t \end{array}$$

$$\Big\downarrow \ \text{Simp}$$

$$\underline{\neg v}, \underline{P(u, u) \approx t} \quad \vdash \quad \begin{array}{r} P(x, y) \not\approx t \lor Q(x) \approx t \lor R(x) \approx t \cdot \emptyset, \\ \underline{Q(x) \approx t \lor R(x) \approx t \cdot P(x, x) \to t} \end{array}$$

$$\Big\downarrow \ \text{Split (left)}$$

$$\begin{array}{r} \neg v, P(u, u) \approx t, \\ Q(u) \approx t \end{array} \quad \vdash \quad \begin{array}{r} P(x, y) \not\approx t \lor Q(x) \approx t \lor R(x) \approx t \cdot \emptyset, \\ Q(x) \approx t \lor R(x) \approx t \cdot P(x, x) \to t \end{array}$$

Derivation Rules (4)

Close applies to

$$C \cdot l_1 \to r_1, \ldots, l_n \to r_n$$

Use conversion to ordinary clause

$$C \vee l_1 \not\approx r_1 \vee \cdots \vee l_n \not\approx r_n$$

and ordinary Close:

$$\text{Close} \ \frac{\Lambda \ \vdash \ \Phi, \ C}{\Lambda \ \vdash \ \bot}$$

Optional Derivation Rules (1)

Assert $$\frac{\Lambda \quad \vdash \quad \Phi}{\Lambda,\ L \quad \vdash \quad \Phi}$$ if L is not subsumed by a context literal and "soundness condition" holds

Examples

No Split for unit clauses:

$$\begin{array}{rrcl} & \Lambda & \vdash & \Phi,\ P(x) \cdot \emptyset \\ \longrightarrow & \Lambda,\ P(x) & \vdash & \Phi,\ P(x) \cdot \emptyset \end{array}$$

With equality reasoning:

$$\begin{array}{rlcl} & P(u,b),\ b \approx c & \vdash & \neg P(x,y) \vee f(x) \approx y \cdot \emptyset \\ \longrightarrow & P(u,b),\ b \approx c,\ f(u) \approx c & \vdash & \neg P(x,y) \vee f(x) \approx y \cdot \emptyset \end{array}$$

Optional Derivation Rules (2)

$$\text{Simp} \quad \frac{\Lambda \quad \vdash \quad \Phi,\ C \cdot \Gamma}{\Lambda \quad \vdash \quad \Phi,\ C' \cdot \Gamma'} \quad \text{if} \begin{cases} C \cdot \Gamma \text{ is redundant wrt.} \\ \quad \Phi \cup \{C' \cdot \Gamma'\} \text{ and } \Lambda\text{, and} \\ \text{``Soundness condition''} \end{cases}$$

Examples

Delete a clause whose constraint will never be satisfied:

$$\begin{array}{lllll} & f(x) \not\approx x & \vdash & a \approx b \cdot f(a) \to a \\ \longrightarrow & f(x) \not\approx x & \vdash & t \approx t \cdot \emptyset \end{array}$$

Simplify constraint:

$$\begin{array}{lllll} & f(x) \approx x & \vdash & a \approx b \cdot f(a) \to a \\ \longrightarrow & f(x) \approx x & \vdash & a \approx b \cdot \emptyset \end{array}$$

Generic Simp rule covers most simplification rules so far as special cases

Contents

- DPLL as a starting point for the Model Evolution calculus
- Model Evolution calculus without equality
  - Model construction
  - Inference rules
- **Equality reasoning**
  - Equality reasoning in instance based methods
  - Inference rules
  - **How it works (semantical considerations)**

Model Evolution Calculus with Equality – How it Works

Without Equality

- Current context Λ
- Candidate Model I_Λ
- Current clause C
- If $I_\Lambda \not\models C$ then repair I_Λ (Split) or give up I_Λ (Close)

With Equality

- Current context Λ
- Candidate E-model R_Λ -- a ground rewrite system
- Current clause C
- If $R_\Lambda \not\models_E C$ then repair R_Λ (Split) or give up R_Λ (Close)

R_Λ **?**

E-Interpretation Induced by a Branch

Initially $R_\Lambda := \emptyset$

For all $s \approx t \in I_\Lambda$, smaller equations first:

  if $s \succ t$ and
    s and t are irreducible by smaller rules in R_Λ
  then $R_\Lambda := R_\Lambda \cup \{s \to t\}$

Example

Candidate equations $I_\Lambda = \{a \dot{\approx} b, b \dot{\approx} c\}$

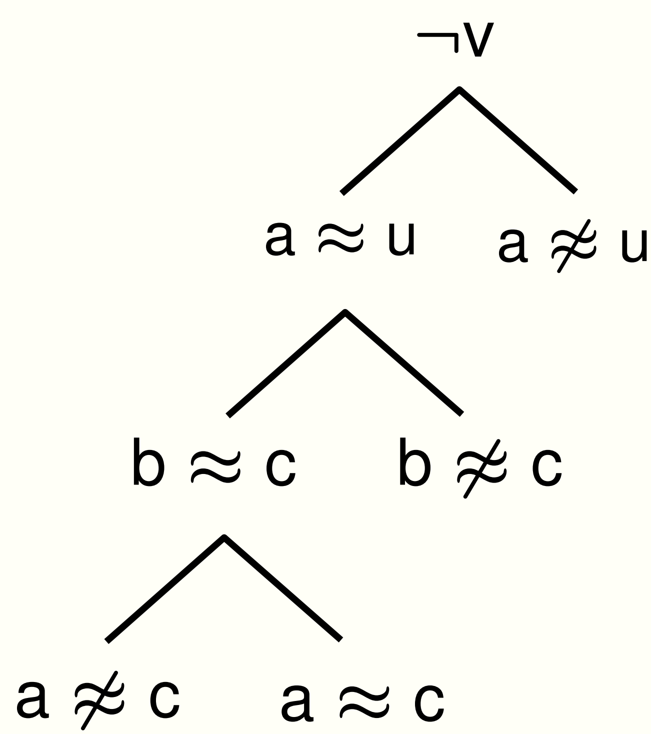
Ordering $a \succ b \succ c$

(1) $R_\Lambda := \emptyset$

(2) candidate $b \approx c$: $R_\Lambda := \{b \to c\}$

(3) candidate $a \approx b$: $R_\Lambda = \{b \to c\}$

Important: R_Λ is convergent



Repairing the Candidate Model

$$R_\Lambda \not\models_E \Phi, C \quad \longrightarrow \quad R_\Lambda \not\models_E C\gamma \quad \longrightarrow \quad R_\Lambda \not\models_E C\gamma \downarrow R_\Lambda$$

$$s \neq t$$

$$\downarrow$$

$$C\gamma \downarrow R_\Lambda = q \not\approx q \vee \cdots \vee s \approx t \vee \cdots$$

Used rewrite rules from R_Λ:
$l \rightarrow r, \ldots$

$\downarrow$ lifted versions

Calculus:

$$C \cdot \emptyset \Rightarrow^{\star}_{\text{Para,Ref}} \quad s' \approx t' \vee \cdots \cdot l' \rightarrow r', \ldots$$

Split with $s' \approx t'$ to add $s \approx t$ to R_Λ, or

Split with $l' \not\approx r'$ to remove $l \rightarrow r$ from R_Λ

(Ground) constrained clauses semantics:

$$R_\Lambda \models_E C \cdot \Gamma \text{ iff } \Gamma \not\subseteq R_\Lambda \text{ or } R_\Lambda \models_E C$$

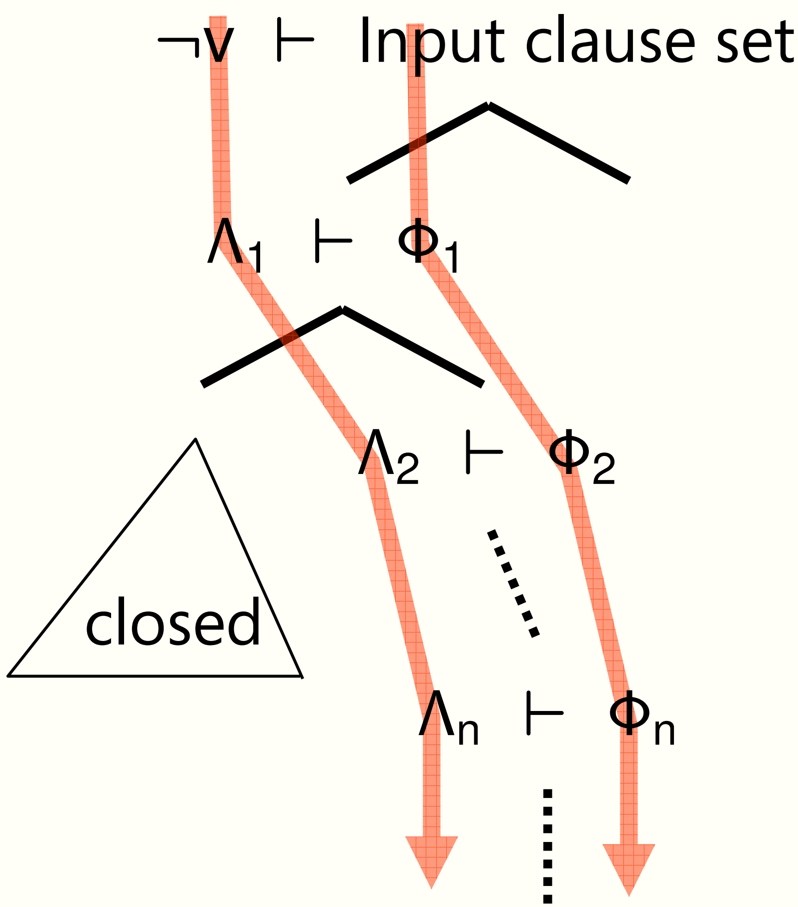
After Split $C\gamma \downarrow R_\Lambda$ will be E-satisfied, and so will be $C\gamma$

Model Construction Considerations

- The model construction technique has been developed for the Superposition calculus [Bachmair&Ganzinger]
- Differences due to parametric literals
  - Nonmonotonicity:

    e.g. $f(u) \approx u$ later partially retracted due to $f(a) \not\approx a$
  - Have to work with **two** orderings:
    term ordering and instantiation ordering
  - Model construction:
    smaller sides of equations must be irreducible, too,
    in order to be turned into rewrite rules
  - In consequence, paramodulation into smaller sides is necessary (really?)

Limit Derivations



$$\Lambda_\infty := \bigcup_{i \geq 0} \bigcap_{j \geq i} \Lambda_j$$

$$\Phi_\infty := \bigcup_{i \geq 0} \bigcap_{j \geq i} \Phi_j$$

Limit rewrite system R_{Λ_∞}

- This is the intended model
- Approximations used in redundancy tests

Completeness

Suppose a fair derivation that is not a closed tree

Show that $R_{\Lambda_\infty} \models \Phi_\infty$

Fairness?

Fairness

Def. (Fairness)

Para Suppose timepoint i in derivation such that

$$\text{Para}\ \frac{\Lambda_i,\ l \approx r \quad \vdash \quad \Phi_i,\ C \cdot \Gamma}{\Lambda_i,\ l \approx r \quad \vdash \quad \Phi_i,\ C \cdot \Gamma,\ C' \cdot \Gamma'}$$

where $C \cdot \Gamma \Rightarrow_{\text{Para}(l \approx r, \boldsymbol{\sigma})} C' \cdot \Gamma'$

If

1. $l \approx r \in \Lambda_B$,
2. Λ_B produces $(l \approx r)\boldsymbol{\sigma}$, and
3. $(C \cdot \Gamma)\boldsymbol{\sigma}$ is not redundant wrt. $\Phi_i \cup \{C \cdot \Gamma\}$ and R_{Λ_B}

then

there is a j such that the inference

$C \cdot \Gamma \Rightarrow_{\text{Para}(l \approx r, \boldsymbol{\sigma})} C' \cdot \Gamma'$ is redundant wrt. Φ_j and R_{Λ_B}

Split, Ref, Close . . .

Conclusions

- Main result: soundness and refutational completeness
- Nice features (perhaps):
  - Paramodulation only from **unit** equations
  - No paramodulation inferences between context equations or into constraint part
  - Clause part of constrained clauses does not grow in length (decide Bernays-Schoenfinkel clauses with equality)
  - Works with explicitly represented model candidate at the calculus level (the context)
- Not so nice features (perhaps):
  - Semantic redundancy criterion based on model candidate difficult to exploit
  - Need paramodulation into smaller sides of equations (really?)