The Model Evolution Calculus with Equality

Peter Baumgartner

Max-Planck-Institut for Computer Science

Saarbrücken, Germany

Cesare Tinelli

The University of Iowa

Iowa, USA

Background and Motivation

- Recently increased interest in **Instance Based Methods**
  - Ordered Semantic Hyper Linking [Plaisted et al], Primal Partial Instantiation [Hooker et al], Disconnection Method [Billon], DCTP [Letz&Stenz], First-Order DPLL [B.], Model Evolution [B.&Tinelli], Inst-Gen [Ganzinger&Korovin]
  - 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)
- Inference rules for equality in Instance Based Methods
  - So far only for Inst-Gen and DCTP
  - How to add equality handling to Model Evolution?

Contents

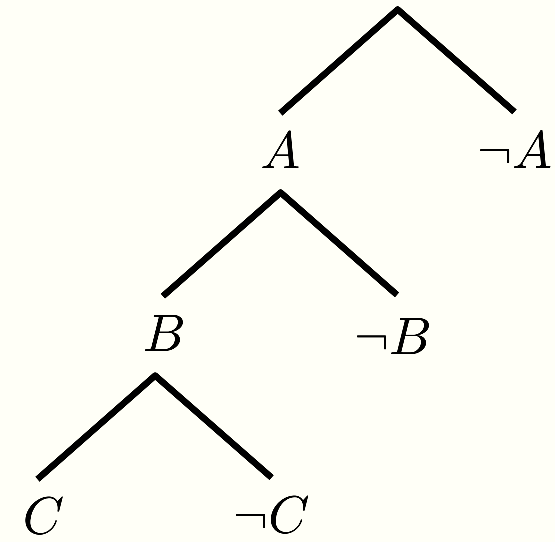
- DPLL as a starting point for the Model Evolution calculus
- Model Evolution calculus
- Adding inference rules for equality

DPLL - Idea

Davis-Putnam-Logemann-Loveland Procedure (1960-63)
Propositional core:

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

Semantic tree enumerates interpretations:



Algorithm components:
- Simplification
- Split
- Backtracking

$$A, B \vdash \cancel{\neg A} \vee \cancel{\neg B} \vee C \vee D, \ldots$$

$$\{A, B\} \overset{?}{\models} \neg A \vee \neg B \vee C \vee D$$

No, split on C:

$$A, B, C \vdash \cancel{\neg A \vee \neg B \vee C \vee D}, \ldots$$

Beyond DPLL

- Recent research in propositional satisfiability (SAT) has been very successful.
- The best modern SAT solvers (satz, MiniSat, zChaff, Berkmin,...) are based on DPLL.
- Can DPLL be lifted to the first-order level?
- Can we combine successful SAT techniques
  (unit propagation, backjumping, learning,...)
  with successful first-order techniques?
  (unification, subsumption, ...)?

Model Evolution Calculus – Idea

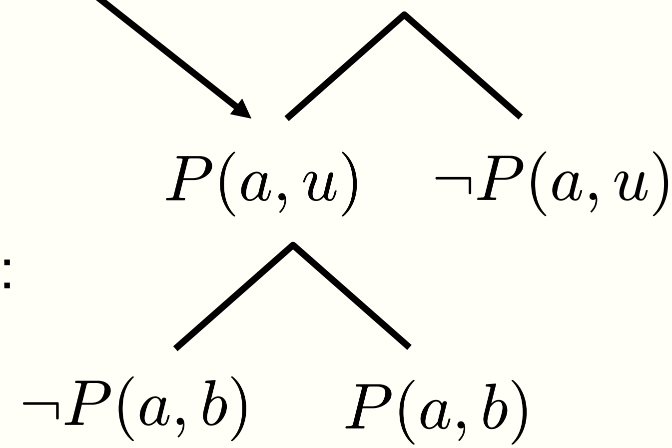
Lifing of tree data structure and derivation rules to first-order

Input: First-order clause set

Output: Model or „unsatisfiable" if termination

u is a parameter,
a special kind of variable

Semantic tree enumerates interpretations:



Procedure components:
- Simplification
- Split
- Backtracking

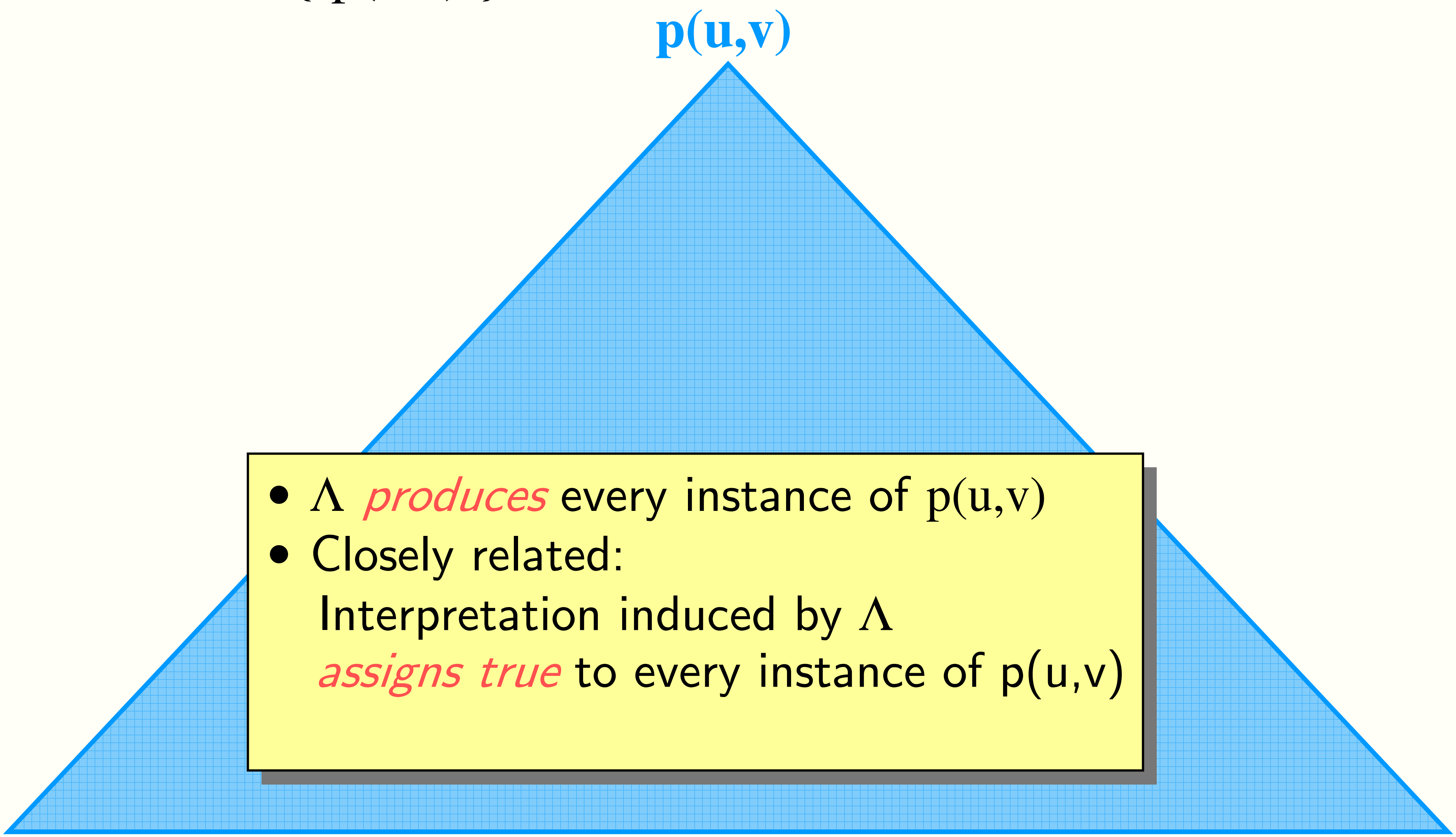
$$P(a,u), \neg P(a,b) \ \vdash \ Q(x,y) \vee P(x,y), \ldots$$

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

Interpretation induced by a branch?

Interpretation Induced by a Branch: Examples

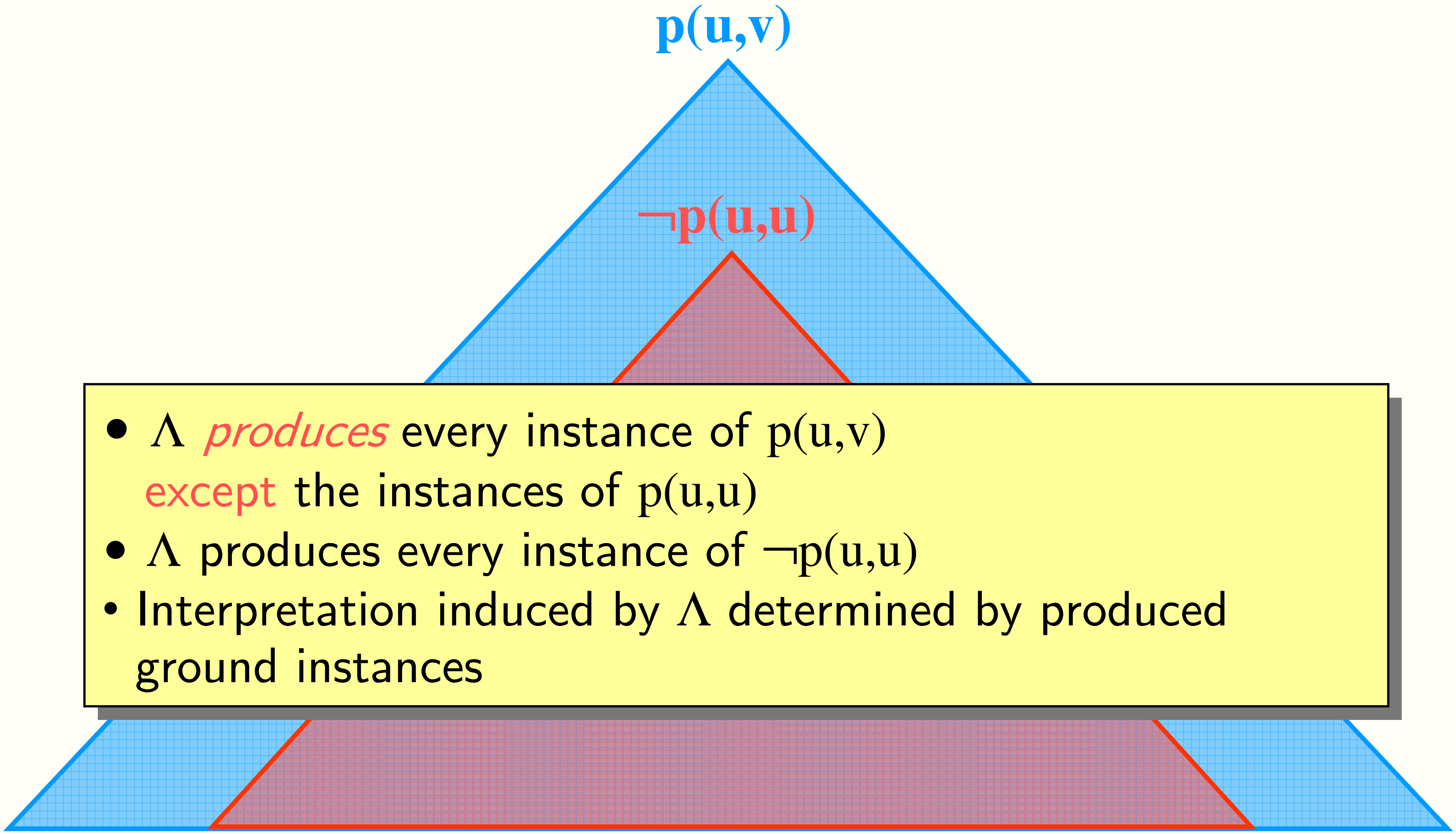
Branch Λ = { p(u,v) }



- Λ *produces* every instance of p(u,v)
- Closely related:
  Interpretation induced by Λ
  assigns true to every instance of p(u,v)

Interpretation Induced by a Branch: Examples

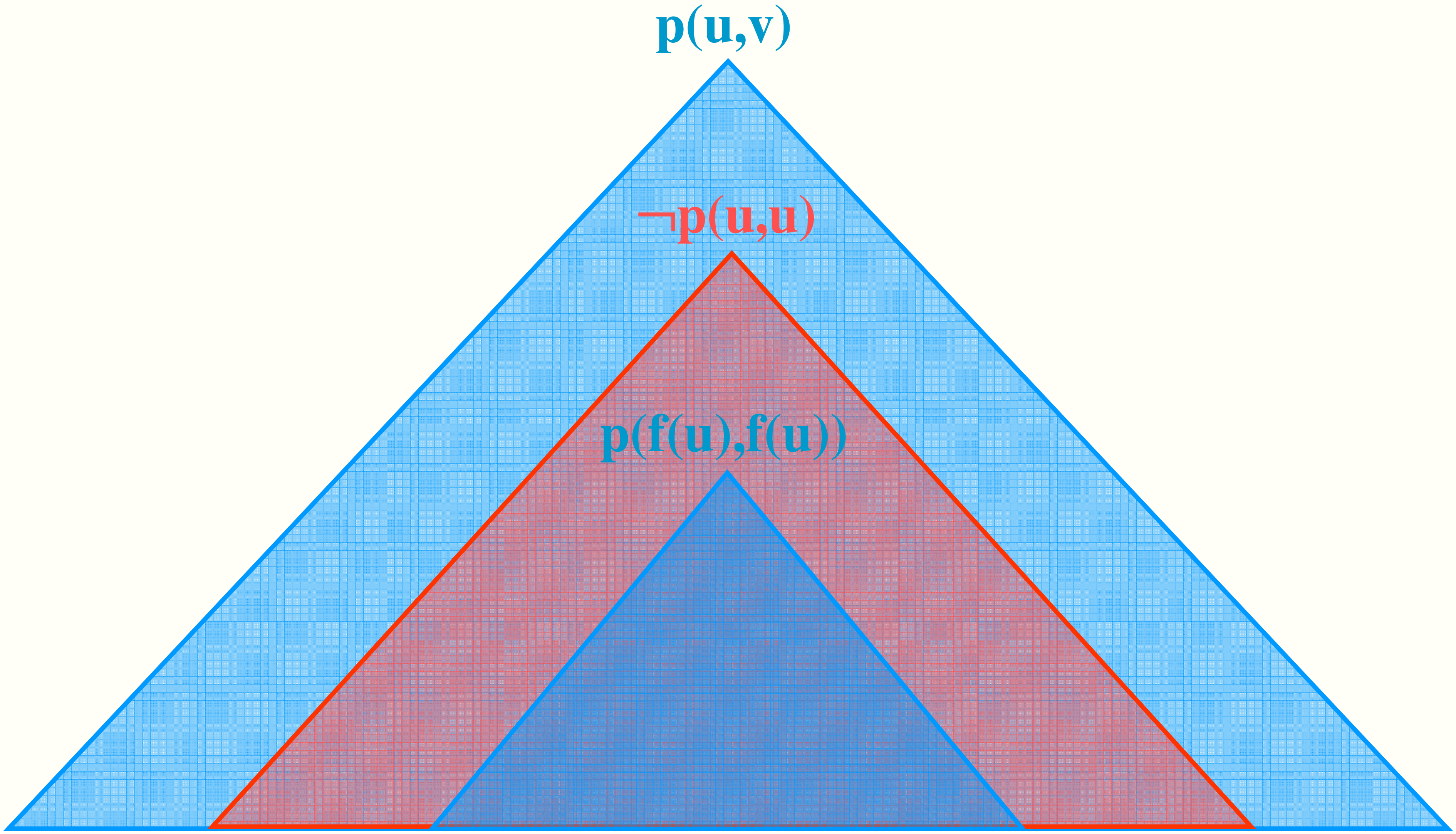
$\Lambda = \{p(u,v), \neg p(u,u)\}$



- Λ *produces* every instance of $p(u,v)$
  except the instances of $p(u,u)$
- Λ produces every instance of $\neg p(u,u)$
- Interpretation induced by Λ determined by produced ground instances

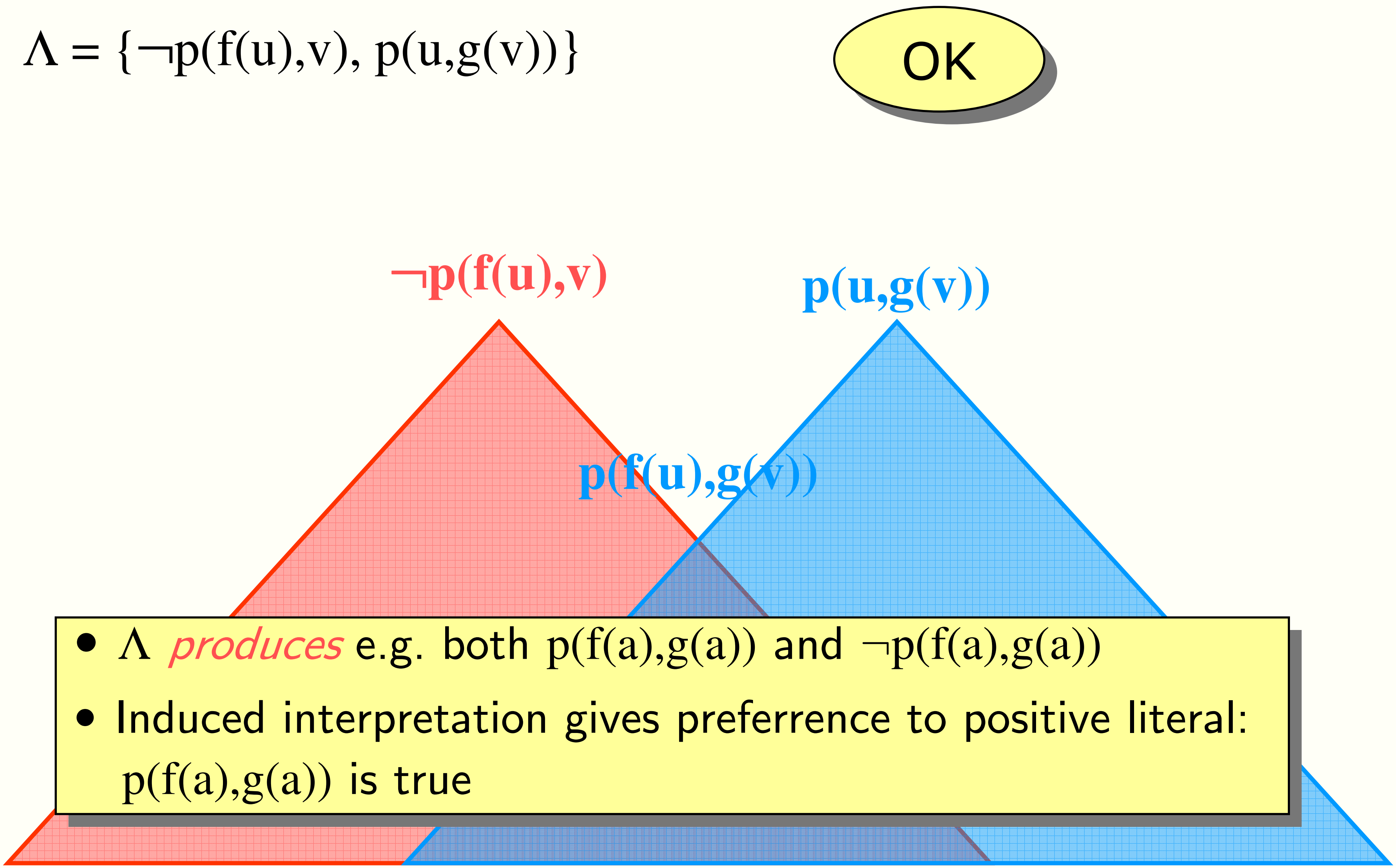
Interpretation Induced by a Branch: Examples

$\Lambda = \{p(u,v), \neg p(u,u), p(f(u),f(u))\}$



Interpretation Induced by a Branch: Examples

$\Lambda = \{\neg p(f(u),v),\ p(u,g(v))\}$

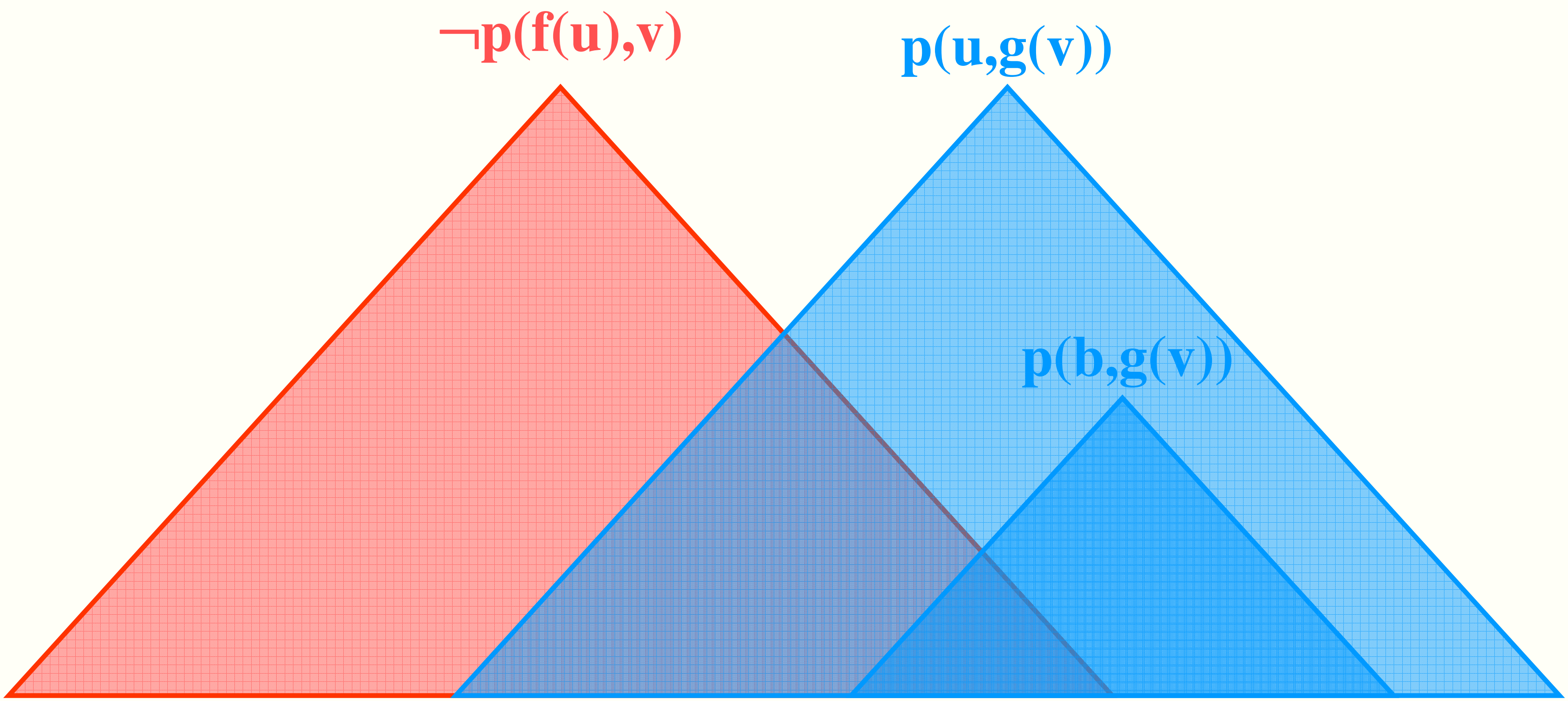


- Λ *produces* e.g. both $p(f(a),g(a))$ and $\neg p(f(a),g(a))$
- Induced interpretation gives preferrence to positive literal: $p(f(a),g(a))$ is true

Interpretation Induced by a Branch: Examples

$\Lambda = \{\neg p(f(u),v),\ p(u,g(v)),\ p(b,g(v))\ \}$

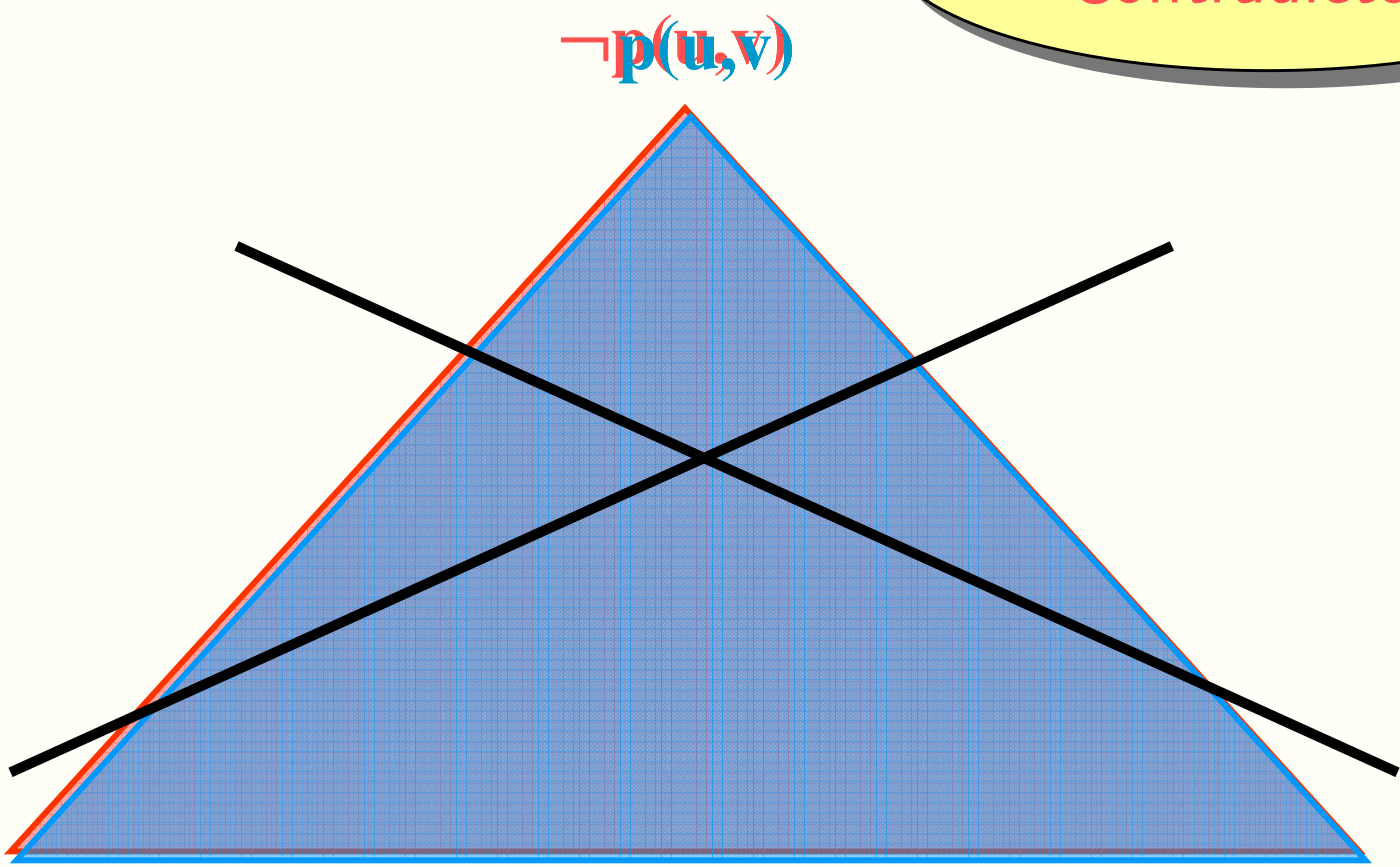
OK



Interpretation Induced by a Branch: Examples

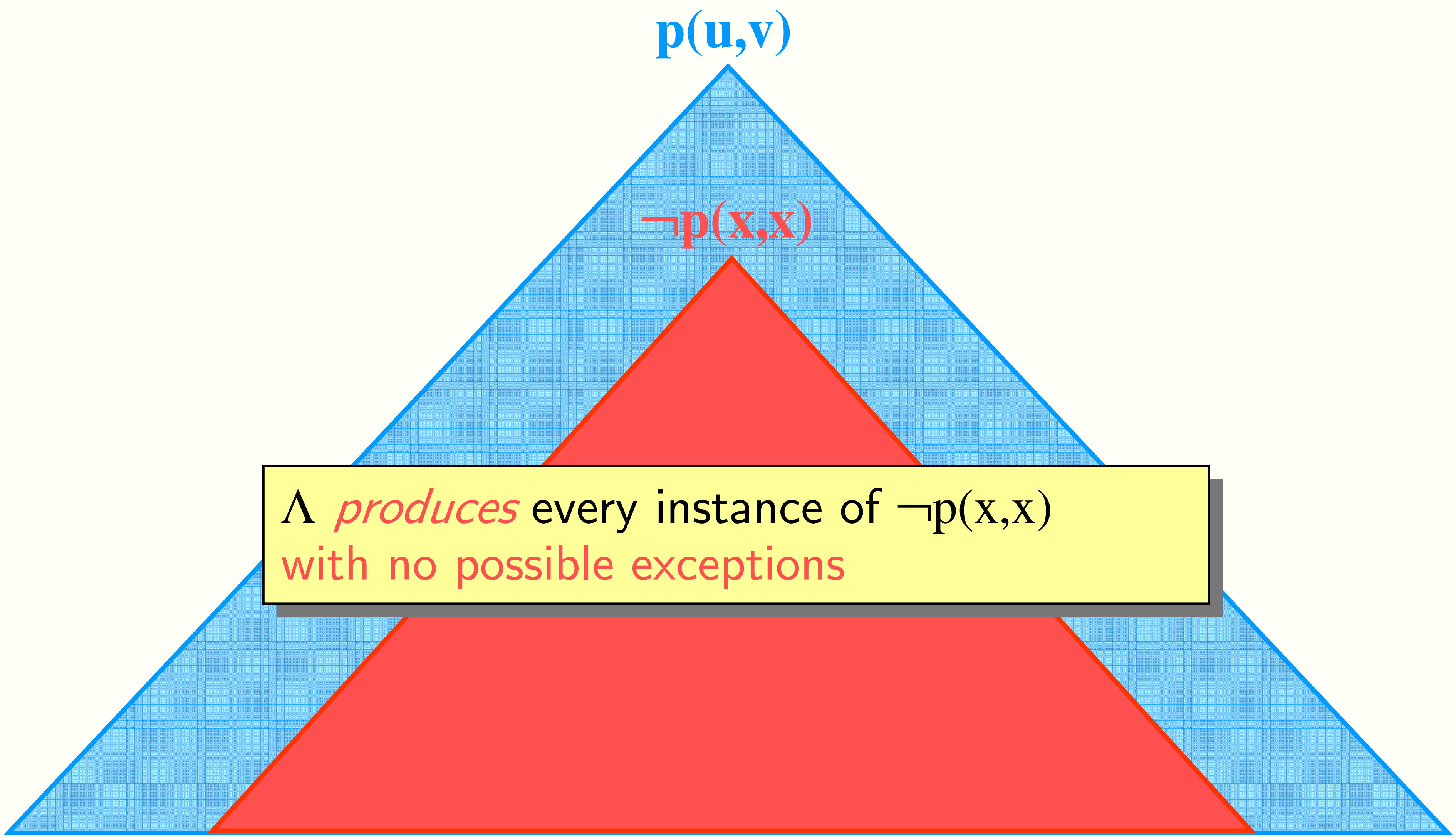
$\Lambda = \{\neg p(u,v),\ p(u,v)\}$

Not OK!
Contradictory



Interpretation Induced by a Branch: Examples

$\Lambda = \{p(u,v), \forall x\ \neg p(x,x)\}$

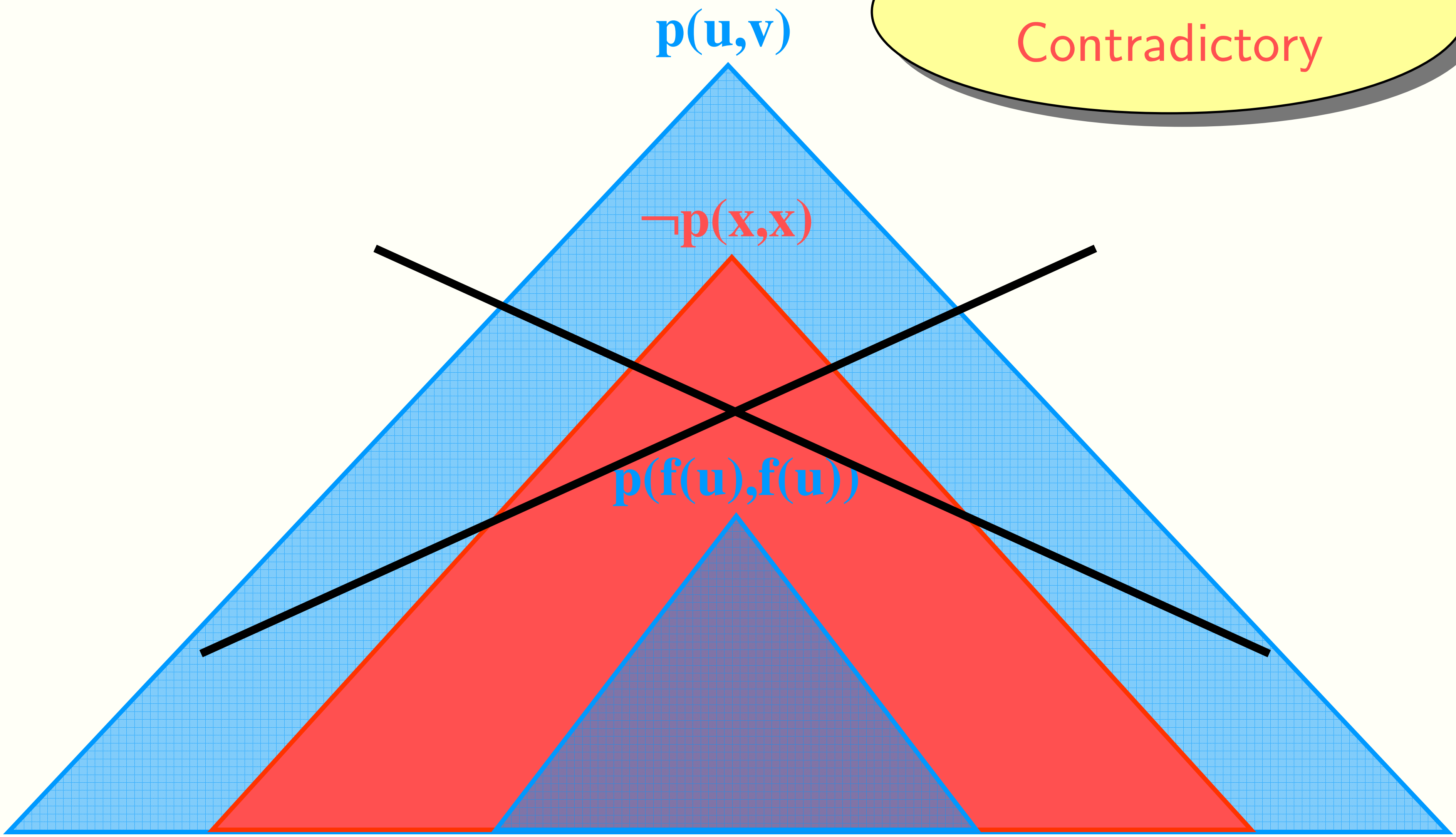


Λ *produces* every instance of $\neg p(x,x)$
with no possible exceptions

Interpretation Induced by a Branch: Examples

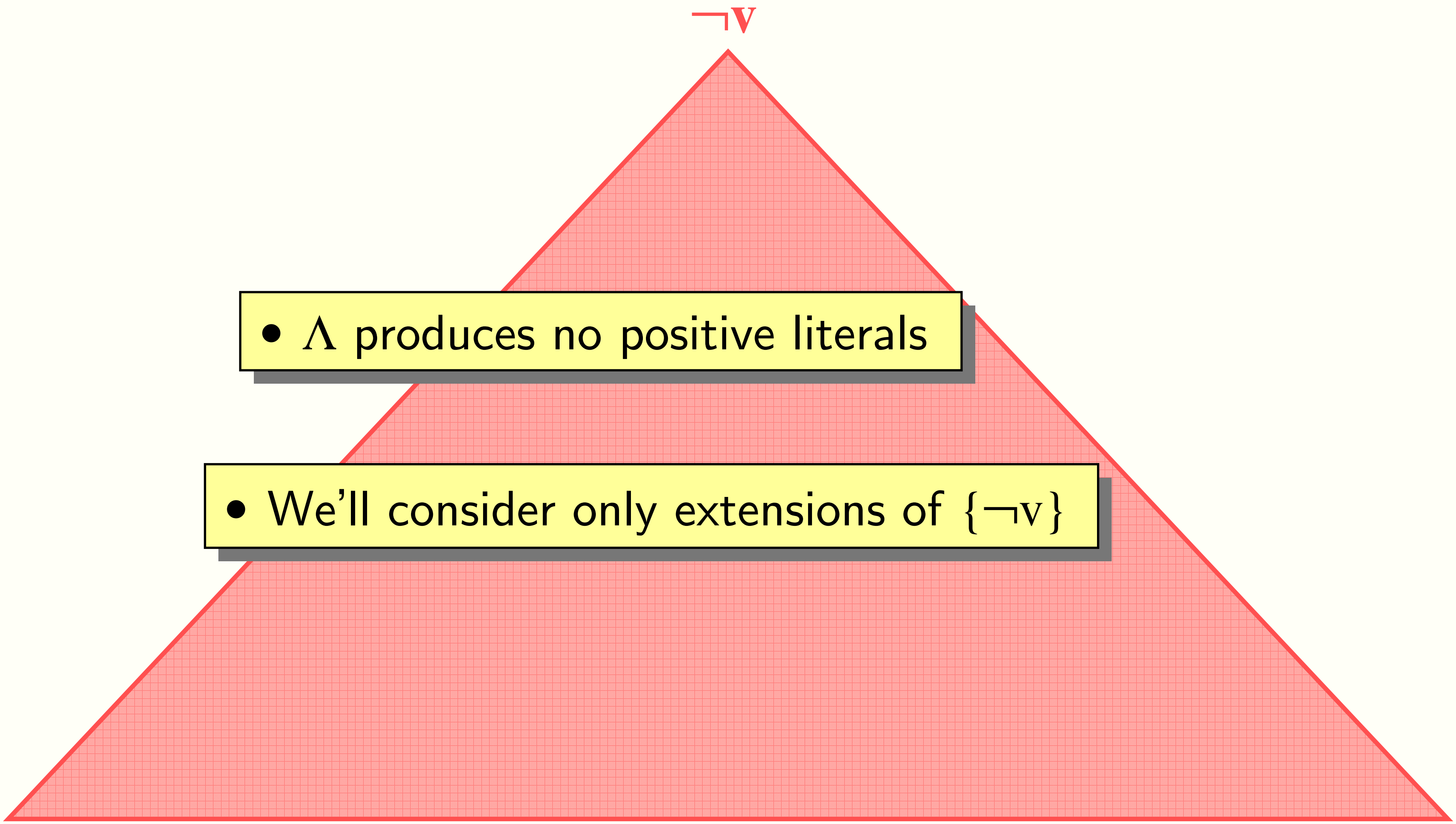
$\Lambda = \{p(u,v), \neg p(x,x), p(f(u),f(u))\}$

Not OK!
Contradictory



Initial Context

$\Lambda = \{\neg v\}$

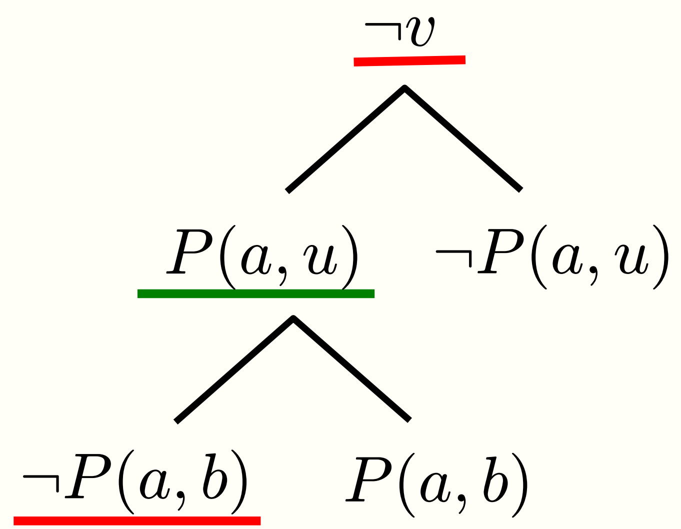


Model Evolution Calculus – Idea (cont'd)

Lifting of semantic trees and derivation rules to first-order

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

Induced Interpretation I_Λ:
 true: $P(a,a)$
 false: $P(a,b), Q(a,b)$



Therefore $\{\neg v, P(a,u), \neg P(a,b)\} \not\models Q(a,b) \vee P(a,b)$

Therefore $\{\neg v, P(a,u), \neg P(a,b)\} \not\models Q(x,y) \vee P(x,y)$

Split with $Q(a,b)$ to satisfy $P(a,b) \vee Q(a,b)$

How to determine Split literal? Calculus?

Derivation Rules – Simplified (1)

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

if

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

$$\Lambda: \quad \underline{P(u,u)} \qquad \underline{Q(v,\mathrm{b})}$$

$$C \vee L : \neg P(x,y) \vee \neg Q(\mathrm{a},z)$$

$$(C \vee L)\sigma : \underline{\neg P(x,x)} \vee \underline{\neg Q(\mathrm{a},\mathrm{b})}$$

$$\sigma = \{\ x \mapsto u,\ y \mapsto u,\ v \mapsto \mathrm{a},\ z \mapsto \mathrm{b}\ \}$$

2. violated   2. satisfied

$L\sigma = \neg Q(\mathrm{a},\mathrm{b})$ is admissible for Split

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 σ of C against Λ such that Λ contains the complement of each literal of $C\sigma$

Λ: $\quad \underline{P(u,u)} \qquad \underline{Q(\mathrm{a},\mathrm{b})}$

$C : \neg P(x,y) \vee \neg Q(\mathrm{a},z)$

$\sigma = \{\, x \mapsto u,\ y \mapsto u,\ z \mapsto \mathrm{b} \,\}$

$C\sigma : \neg \underline{P(x,x)} \vee \neg \underline{Q(\mathrm{a},\mathrm{b})}$

2. satisfied 2. satisfied

Close is applicable

Derivation Rules – Simplification Rules (1)

Propositional level:

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

First-order level $\approx$ unit subsumption:

- All variables in context literal L must be universally quantified
- Replace equality by matching

Derivation Rules – Simplification Rules (2)

Propositional level:

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

First-order level $\approx$ restricted unit resolution

- All variables in context literal L must be universally quantified
- Replace equality by unification
- The unifier must not modify C

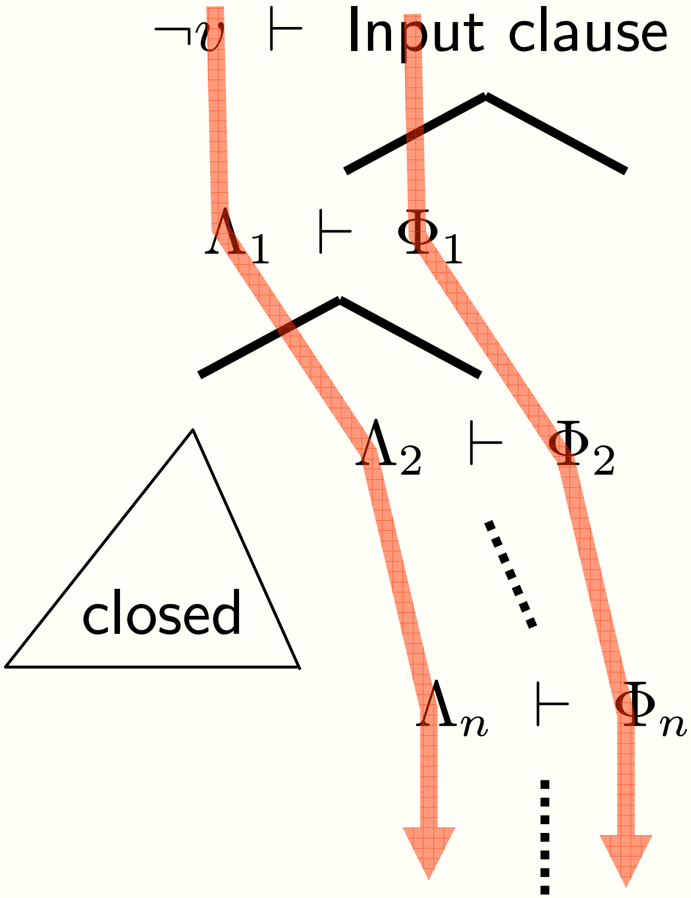
Derivation Rules – Simplification Rules (3)

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

if

1. all variables in K are universally quantified
2. $K\sigma = L$, for some substitution σ

Derivations and Completeness



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

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

Fairness

Closed tree or open limit tree,
with some branch satisfying:

1. Close not applicable to any Λ_i
2. For all $C \in \Phi_\infty$ and subst. γ,
   "if for some i, $\Lambda_i \not\models C\gamma$
   then there is $j \geq i$
   such that $\Lambda_j \models C\gamma$

   (Use Split to achieve this)

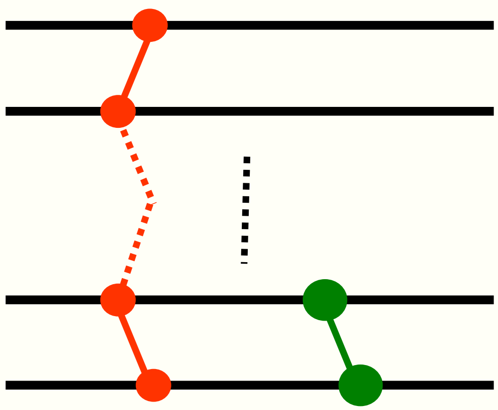
Completeness

Suppose a fair derivation
that is not a closed tree

Show that $\Lambda_\infty \models \Phi_\infty$

Equality

- Design decision:
  - „Total theory reasoning“
    a la [Ganzinger&Korovin]
    OR
    „Partial theory reasoning“
    a la [Letz&Stenz]



  - Our approach:
    Partial theory reasoning (Paramodulation)
- Generalizations
  - Branch represents E-Interpretation now
  - Paramodulation inference rules
  - Redundancy

Will talk about generalizations in this order

E-Interpretation Induced by a Branch

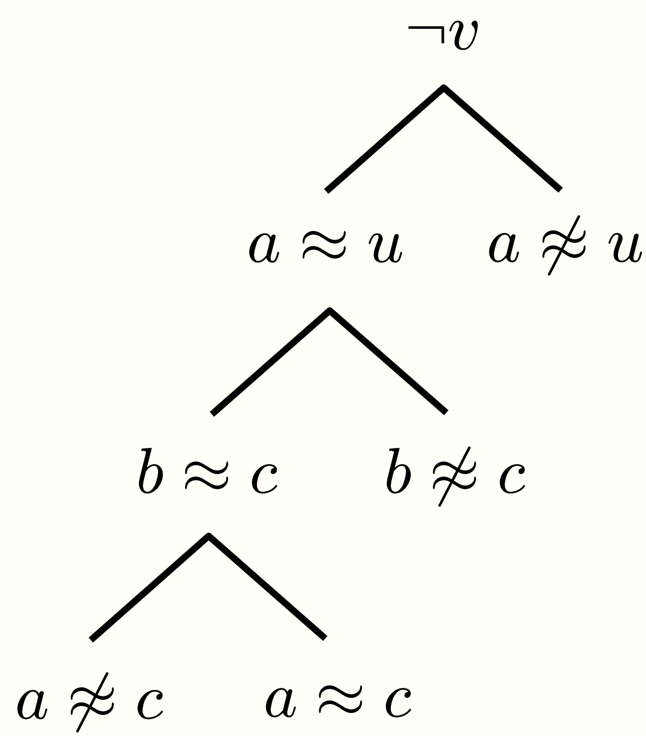
Obtain I_Λ as produced positive ground equations, modulo symmetry

E.g. for left branch

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

Induced E-Interpretation presented as a rewrite system, here

$$R_\Lambda = \{b \to c\}$$



Construction of the Rewrite System

E.g. $a \succ b \succ c$ induces

$$a \approx b \succ a \approx c \succ b \approx c$$

Take reduction ordering $\succ$

Set initial rewrite system $R_\Lambda := \emptyset$

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

$R_\Lambda := R_\Lambda \cup \{s \to t\}$ if $s \succ t$ and

s and t are irreducible wrt. smaller rules in R_Λ

E-Interpretation Induced by a Branch (cont‘d)

- The model construction technique has been developed for the Superposition calculus [Bachmair&Ganzinger]
- Relevant here in particular special case of unit clauses
- Differences due to parametric literals
  - Nonmonotonicity:
    e.g. $f(u) \approx u$ later partially retracted by $f(a) \not\approx a$
  - 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 neccessary (really?)

Paramodulation and Constrained Clauses

Calculus Idea

By paramodulation derive lifted version of
R_Λ-normalform of a ground instance falsified by R_Λ

Then apply Close or Split

Paramodulation inference, first attempt (unsound)

$$\underline{a \approx u},\ a \not\approx c,\ \neg P(c)\ \vdash\ \underline{P(a)}$$

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

$$a \approx u,\ a \not\approx c,\ \underline{\neg P(c)}\ \vdash\ P(a),\ \underline{P(x)}$$

$$\Big\downarrow \text{Close}$$

$$a \approx u,\ a \not\approx c,\ \neg P(c)\ \vdash\ P(a),\ P(x),\ \Box$$

Paramodulation and Constrained Clauses

Paramodulation inference second attempt (sound)

$$\underline{a \approx u},\ a \not\approx c,\ \neg P(c)\ \vdash\ \underline{P(a) \cdot \emptyset}$$

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

$$a \approx u,\ a \not\approx c,\ \neg P(c)\ \vdash\ P(a) \cdot \emptyset,\ P(x) \cdot a \to x$$

$C \cdot \Gamma$ is called **constrained clause** (with constraint Γ)

Semantics: $R_\Lambda \overset{?}{\models} (C \cdot \Gamma)\gamma$ Example: $\{a \to b\} \overset{?}{\models} P(c) \cdot a \to c$

If $\Gamma\gamma \subseteq R_\Lambda$ then $R_\Lambda \overset{?}{\models} C\gamma$

If $\Gamma\gamma \not\subseteq R_\Lambda$ then $(C \cdot \Gamma)\gamma$ was not obtained by rewriting with R_Λ

Because $(C \cdot \Gamma)\gamma$ is irrelevant then define $R_\Lambda \models (C \cdot \Gamma)\gamma$

Basis to define entailment between constrained clause sets and redundant constrained clauses

Derivation Rules (1)

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

if

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

Derivation Rules (2)

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

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 (3)

The Split rule:
applies only to a constrained clause $C \cdot l_1 \to r_1, \ldots, l_n \to r_n$
when C is a positive clause

Read it as an ordinary clause $C \vee l_1 \not\approx r_1 \vee \cdots \vee l_n \not\approx r_n$

Now take Split rule from above:

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

if

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

Note item 3: Variables are OK, just presentation is simplified here

Derivation Rules (4)

The Close rule: as for Split, read a constrained clause
$C \cdot l_1 \to r_1, \ldots, l_n \to r_n$ as an ordinary clause
$C \vee l_1 \not\approx r_1 \vee \cdots \vee l_n \not\approx r_n$

Now take Close rule from above:

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

if

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

Derivation Example

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

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

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

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

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

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

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

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

Optional Derivation Rules (1)

Assert $$\frac{\Lambda \quad \vdash \quad \Phi}{\Lambda,\ L \quad \vdash \quad \Phi}$$ if „see paper“

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 resoning:

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

Optional Derivation Rules (2)

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

Examples

Delete a clause whose constraint will never be satisfied:

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

Simplify constraint:

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

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

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

If

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

then

there is a j such that the inference

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

Split, Ref, Close . . .

Conclusions

- Main result: soundness and refutational completeness
- Nice features (perhaps):
  - Paramodulates only **unit** equations into clauses
  - No paramodulation inferences between context equations or into constrained literals
  - 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?)