The Model Evolution Calculus

Peter Baumgartner, MPI Saarbruecken and U Koblenz

Cesare Tinelli, U Iowa

Background

- Recent research in propositional satisfiability (SAT) has been very successful.
- An effective method for SAT was pioneered by Davis, Putman, Logemann, and Loveland (DPLL).
- The best modern SAT solvers (MiniSat, zChaff, Berkmin,…) are based on DPLL.

The DPLL Procedure: Main Idea

$$\{ \cancel{p \vee q},\ q \vee \neg r \vee s,\ \neg \cancel{p} \vee \neg q,\ \neg \cancel{p} \vee \neg r \vee \neg s,\ \cancel{p} \}$$

$$\Downarrow \quad \textbf{assert}\text{: } p = \mathbf{T}$$

$$\{ q \vee \neg r \vee s,\ \neg q,\ \neg r \vee \neg s \}$$

The DPLL Procedure: Main Idea

$$\{\ p \vee q,\ \ q \vee \neg r \vee s,\ \ \neg p \vee \neg q,\ \ \neg p \vee \neg r \vee \neg s,\ \ p\ \}$$

$$\Downarrow \quad \textbf{assert}\text{: } p = \mathbf{T}$$

$$\{\ \cancel{q} \vee \neg r \vee s,\ \ \cancel{\neg q},\ \ \neg r \vee \neg s\ \}$$

$$\Downarrow \quad \textbf{assert}\text{: } q = \mathbf{F}$$

$$\{\ \neg r \vee s,\ \ \neg r \vee \neg s\ \}$$

The DPLL Procedure: Main Idea

$$\{\ p \vee q,\ \ q \vee \neg r \vee s,\ \ \neg p \vee \neg q,\ \ \neg p \vee \neg r \vee \neg s,\ \ p\ \}$$

$$\Big\Downarrow \quad \textbf{assert}\text{: } p = \mathbf{T}$$

$$\{\ \ q \vee \neg r \vee s,\ \ \neg q,\ \ \neg r \vee \neg s\ \}$$

$$\Big\Downarrow \quad \textbf{assert}\text{: } q = \mathbf{F}$$

$$\{\ \ \cancel{\neg r} \vee s,\ \ \cancel{\neg r} \vee \neg s\ \}$$

$$\Big\Downarrow \quad \textbf{guess}\text{: } r = \mathbf{T}$$

$$\{\ \ s,\ \ \neg s\ \}$$

The DPLL Procedure: Main Idea

$$\{\ p \vee q,\ \ q \vee \neg r \vee s,\ \ \neg p \vee \neg q,\ \ \neg p \vee \neg r \vee \neg s,\ \ p\ \}$$

$$\Downarrow \quad \textbf{assert}\text{: } p = \mathbf{T}$$

$$\{\ \ q \vee \neg r \vee s,\ \ \neg q,\ \ \neg r \vee \neg s\ \}$$

$$\Downarrow \quad \textbf{assert}\text{: } q = \mathbf{F}$$

$$\{\ \ \neg r \vee s,\ \ \neg r \vee \neg s\ \}$$

$$\Downarrow \quad \textbf{guess}\text{: } r = \mathbf{T}$$

$$\{\ \ s,\ \ \neg s\ \}$$

$$\Downarrow$$

contradiction!

The DPLL Procedure: Main Idea

$$\{\ p \vee q,\ \ q \vee \neg r \vee s,\ \ \neg p \vee \neg q,\ \ \neg p \vee \neg r \vee \neg s,\ \ p\ \}$$

$\Downarrow$ **assert**: $p = \mathbf{T}$

$$\{\ \ q \vee \neg r \vee s,\ \ \neg q,\ \ \neg r \vee \neg s\ \}$$

$\Downarrow$ **assert**: $q = \mathbf{F}$

$$\{\ \ \cancel{\neg r \vee s},\ \ \cancel{\neg r \vee \neg s}\ \}$$

$\Downarrow$ **guess**: $r = \mathbf{F}$

$$\{\ \ \}$$

satisfiable!

Correctness of DPLL method

Prop. A formula φ is satisfiable iff there is a sequence of guesses such that $\mathrm{DPLL}(\varphi) = \emptyset$

Research Questions

- Can we lift DPLL to the first-order level?

- Can we combine successful SAT techniques (unit propagation, backjumping, learning,...) with successful first-order techniques (unification, subsumption, ...)?

Previous Work

- **Instance based methods**
  - (O)SHL [Plaisted],
  - Disconnection method [Billon], [Letz, Stenz],
  - Hyper Tableaux Next Generation [Baumgartner],
  - Primal/Dual approach [Hooker et al],
  - Ganzinger-Korovin method

- **First-Order DPLL** [Baumgartner]
  - proper lifting of split rule

This Work

The Model Evolution Calculus

$\approx$

First-Order DPLL

+ DPLL's simplification rules

+ Universal variables

The calculus is a direct lifting of the whole DPLL to the first-order level.

Overview

- The DPLL method as a sequent-style calculus
- A model generation view of DPLL
- The Model Evolution calculus as a lifting of the DPLL calculus
- Properties of the ME calculus
- Further Work

The DPLL Calculus

$$\Lambda = \{literals\} \qquad \Phi = \{clauses\}$$

$$\Lambda \quad \Box \quad \Phi$$

Context
(asserted literals)

The DPLL Calculus

$\Lambda = \{literals\}$ $\Phi = \{clauses\}$

L literal W empty clause C clause

(**subsume**) $\dfrac{\Lambda \quad \square \quad \Phi,\ L \vee C}{\Lambda \quad \square \quad \Phi}$ if $L \in \Lambda$

(**resolve**) $\dfrac{\Lambda \quad \square \quad \Phi,\ L \vee C}{\Lambda \quad \square \quad \Phi,\ C}$ if $\overline{L} \in \Lambda$

(**close**) $\dfrac{\Lambda \quad \square \quad \Phi,\ \mathrm{W}}{\Lambda \quad \square \quad \mathrm{W}}$

The DPLL Calculus

$\Lambda = \{literals\}$ $\Phi = \{clauses\}$

L literal W empty clause C clause

$$(\textbf{subsume}) \quad \frac{\Lambda \quad \square \quad \Phi,\ L \vee C}{\Lambda \quad \square \quad \Phi} \quad \text{if } L \in \Lambda$$

$$(\textbf{resolve}) \quad \frac{\Lambda \quad \square \quad \Phi,\ L \vee C}{\Lambda \quad \square \quad \Phi,\ C} \quad \text{if } \overline{L} \in \Lambda$$

$$(\textbf{close}) \quad \frac{\Lambda \quad \square \quad \Phi,\ L_1 \vee \ldots \vee L_n}{\Lambda \quad \square \quad \mathrm{W}} \quad \text{if } \overline{L_1}, \ldots, \overline{L_n} \in \Lambda$$

The DPLL Calculus (cont.)

$\Lambda = \{literals\}$ $\qquad\qquad$ $\Phi = \{clauses\}$

L literal $\qquad$ W empty clause $\qquad$ C clause

$$(\textbf{assert}) \quad \frac{\Lambda \quad \square \quad \Phi, L}{\Lambda, L \quad \square \quad \Phi, L} \quad \text{if } \begin{cases} L \notin \Lambda \\ \overline{L} \notin \Lambda \end{cases}$$

$$(\textbf{split}) \quad \frac{\Lambda \quad \square \quad L \vee C,\ \Phi}{\Lambda, L \quad \square \quad L \vee C,\ \Phi \qquad \Lambda, \overline{L} \quad \square \quad L \vee C,\ \Phi} \quad \text{if } \begin{cases} C \neq \mathrm{W} \\ L \notin \Lambda \\ \overline{L} \notin \Lambda \end{cases}$$

The DPLL Calculus: Key Insight

$$\Lambda \quad \Box \quad \Phi$$

Λ can be seen as a finite representation of a Herbrand interpretation:

$$I_\Lambda = \{\, L \in \Lambda \mid L \text{ is positive} \,\}$$

If I_Λ does not satisfy Φ,
"repair" it by adding literals to Λ

Some Notation

$I_\Lambda \mathrel{\|\!=} \varphi$ iff $I_\Lambda \models \varphi$ and
there is no $\Lambda' \supseteq \Lambda$ s.t. $I_{\Lambda'} \models \neg\varphi$

permanently satisfies

Examples:

$I_{\{p\}} \models$ ... $\lor \neg q$

Note: at the prop. level

$I_\Lambda \mathrel{\|\!=} L$ iff $L \in \Lambda$

$I_{\{\neg p\}} \models \neg p \lor \neg q$ and $I_{\{\neg p\}} \mathrel{\|\!=} \neg p \lor \neg q$

The DPLL Calculus Revisited: A Model Evolution View

$$(\textbf{subsume})\quad \frac{\Lambda \quad \square \quad \Phi,\ L \vee C}{\Lambda \quad \square \quad \Phi} \quad \text{if } I_\Lambda \Vdash L$$

$$(\textbf{resolve})\quad \frac{\Lambda \quad \square \quad \Phi,\ L \vee C}{\Lambda \quad \square \quad \Phi,\ C} \quad \text{if } I_\Lambda \Vdash \overline{L}$$

$$(\textbf{close})\quad \frac{\Lambda \quad \square \quad \Phi,\ C}{\Lambda \quad \square \quad \mathrm{W}} \quad \text{if } I_\Lambda \Vdash \neg C$$

The DPLL Calculus Revisited: A Model Evolution View

$$(\textbf{split}) \quad \frac{\Lambda \quad \Box \quad L \vee C,\ \Phi}{\Lambda, L \quad \Box \quad L \vee C,\ \Phi \qquad \Lambda, \overline{L} \quad \Box \quad L \vee C,\ \Phi} \quad \text{if } (*)$$

$$(*) \quad = \quad \begin{cases} 1) \ \text{not } \ I_{\Lambda} \models L \vee C \\ 2) \ L \text{ not contradictory with } \Lambda \\ 3) \ \overline{L} \text{ not contradictory with } \Lambda \end{cases}$$

Note: $(*)$ implies $\begin{cases} I_{\Lambda,L} \Vdash L & \qquad I_{\Lambda,L} \Vdash L \vee C \\ I_{\Lambda,\overline{L}} \Vdash \overline{L} & \qquad \text{not } \ I_{\Lambda,\overline{L}} \models L \vee C \end{cases}$

The DPLL Calculus Revisited: A Model Evolution View

$$(\textbf{assert})\quad \frac{\Lambda \quad \Box \quad \Phi, L}{\Lambda, L \quad \Box \quad \Phi, L} \quad \text{if } (*)$$

$$(*) \quad = \quad \begin{cases} \text{not } \; I_{\Lambda} \Vdash L \\ L \text{ not contradictory with } \Lambda \end{cases}$$

Lifting DPLL to First Order Logic

Main questions:

- How to use contexts to represent a FOL Herbrand interpretation
- What is a contradictory context
- How to check $\models$
- How to check $\Vdash$
- How to repair an interpretation

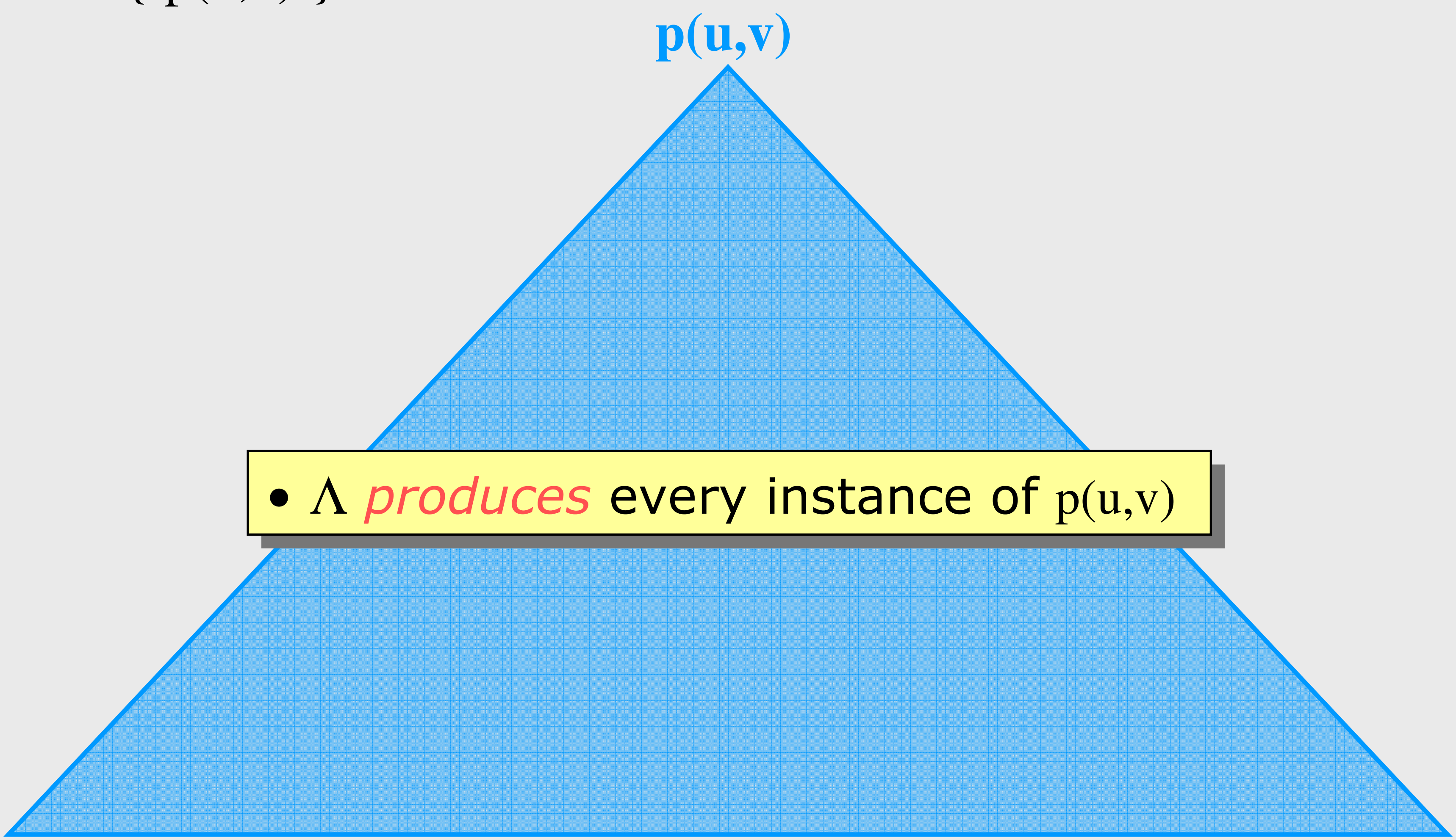
First-order Contexts

Sets Λ of parametric literals $L(u,v,..)$ and
universal literals $L(x,y,\ldots)$

- parameters $(u,v, \ldots)$ and variables $(x,y,\ldots)$ both stand for ground terms
- (roughly) a parametric literal L in Λ denotes all of its ground instances, unless $\neg L' \in \Lambda$ for some instance L' of L
- a universal literal denotes all of its ground instances, unconditionally

First-order Contexts: Examples

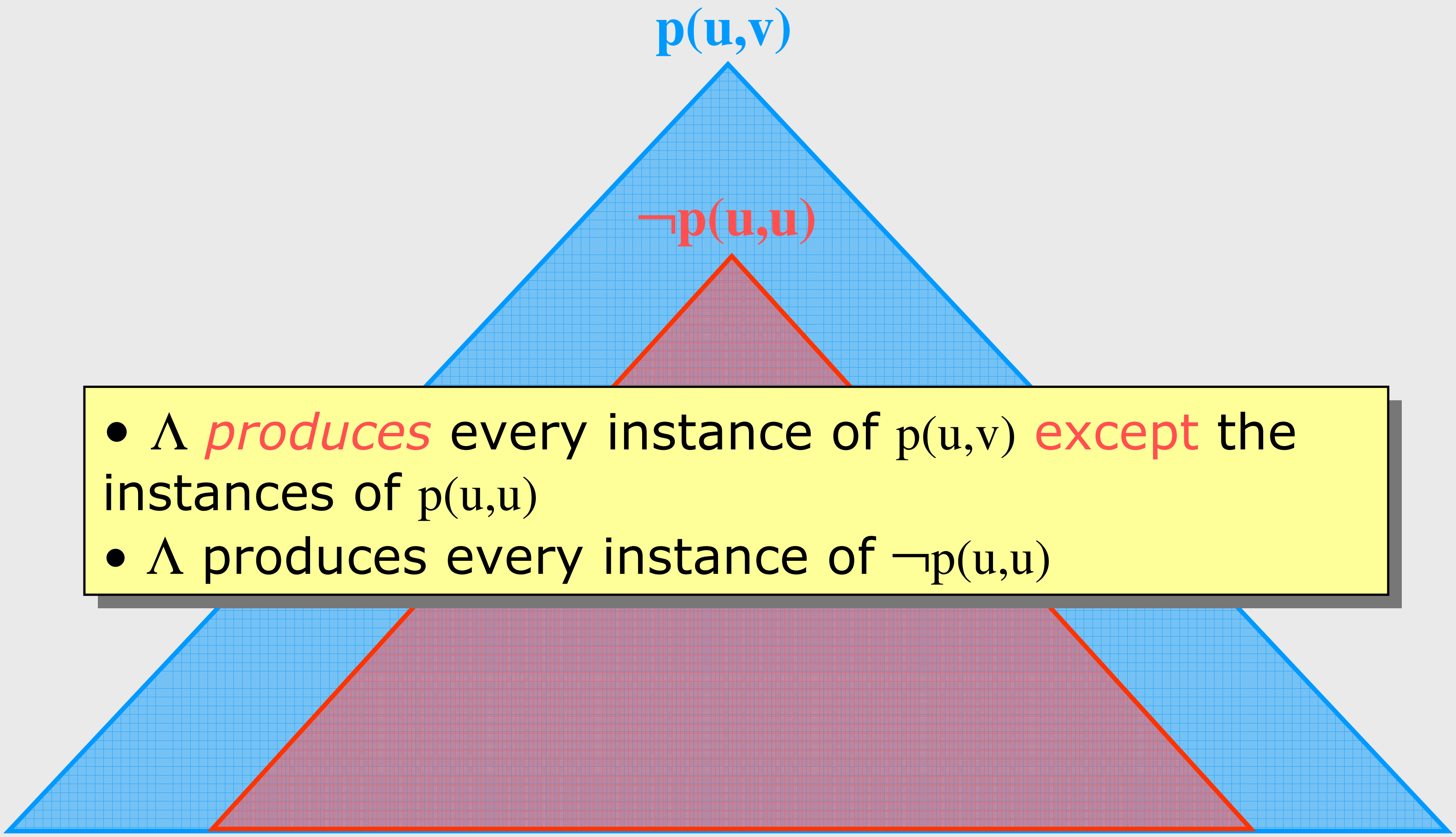
$\Lambda = \{\ p(u,v)\ \}$



• Λ *produces* every instance of $p(u,v)$

First-order Contexts: Examples

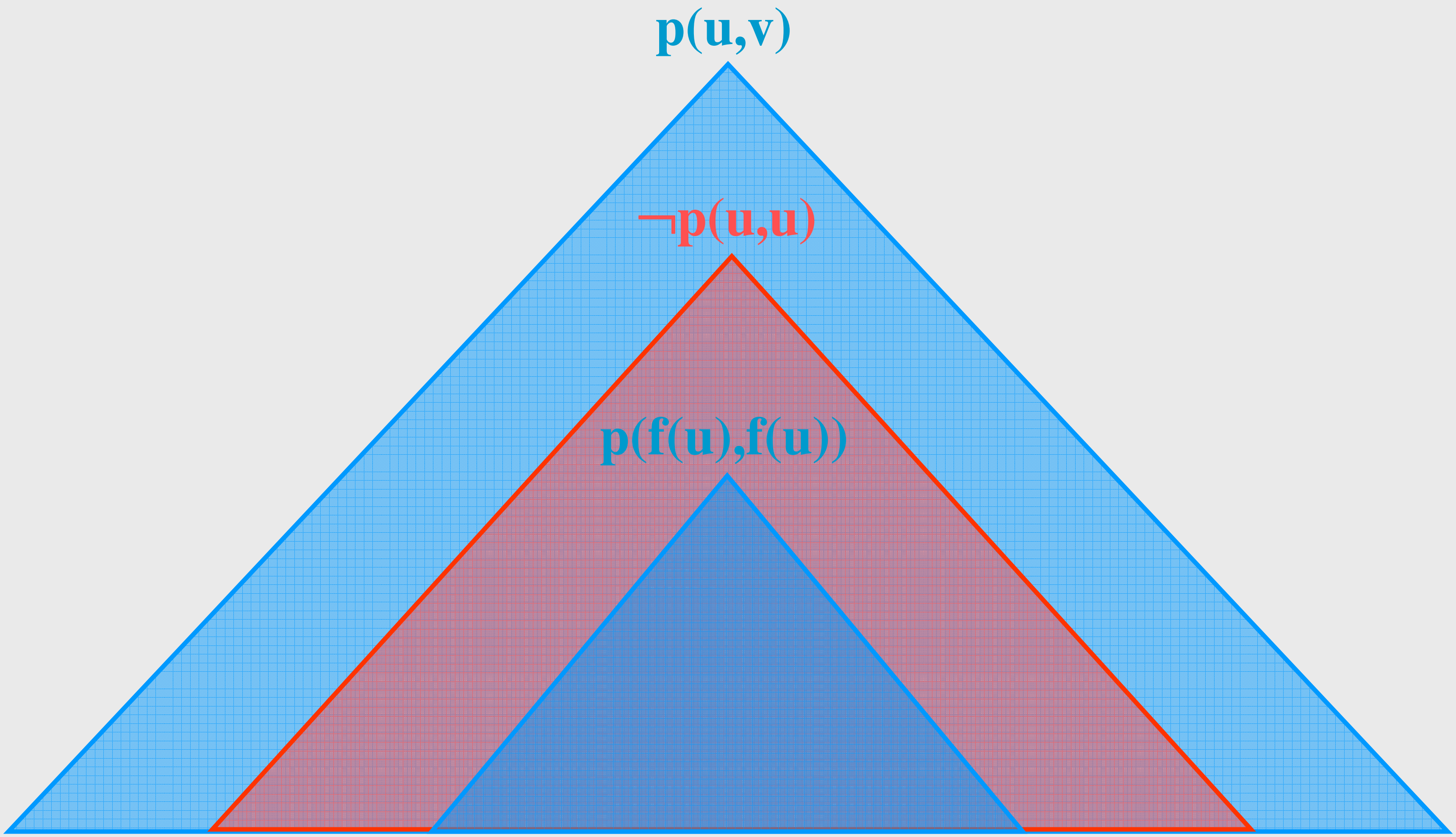
$\Lambda = \{p(u,v), \neg p(u,u)\}$



- Λ *produces* every instance of $p(u,v)$ except the instances of $p(u,u)$
- Λ produces every instance of $\neg p(u,u)$

First-order Contexts: Examples

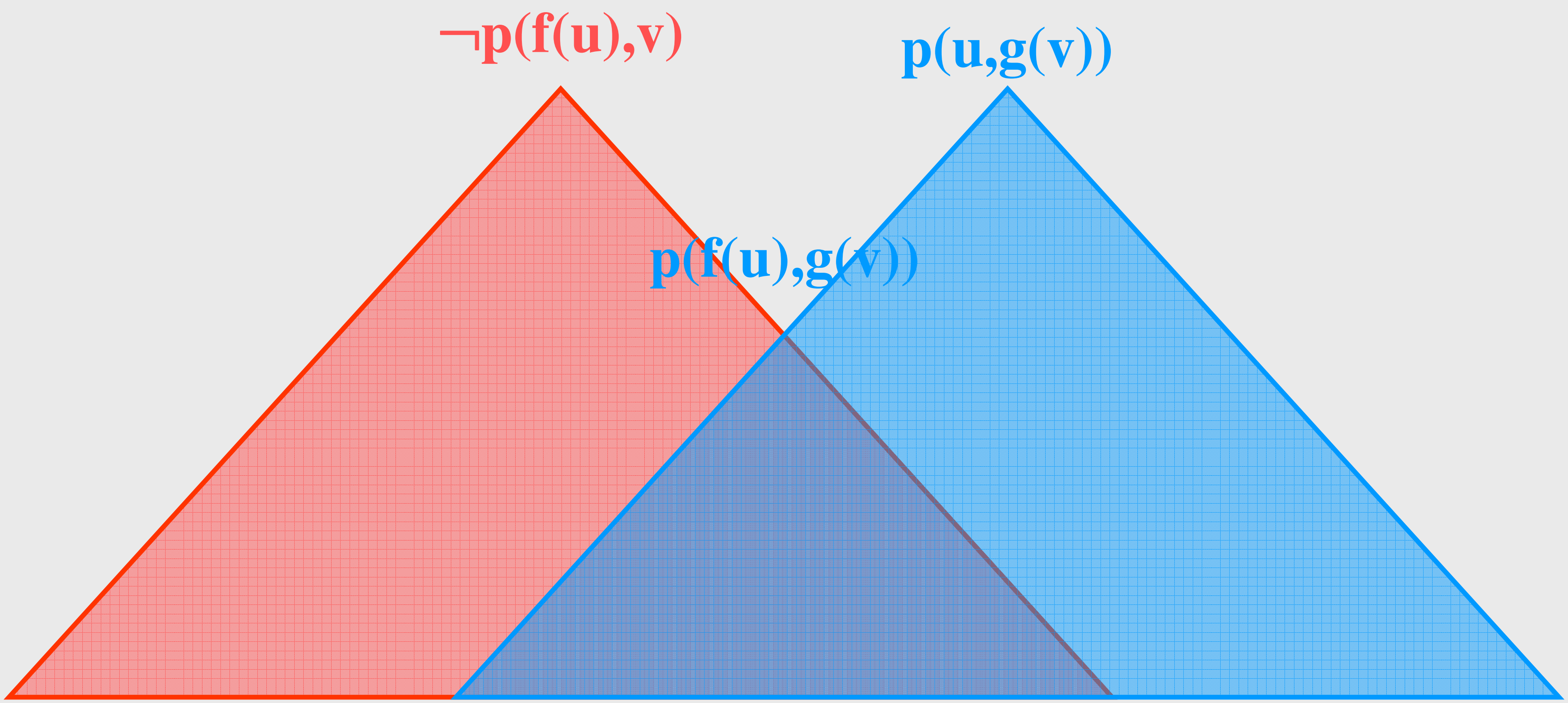
$\Lambda = \{p(u,v), \neg p(u,u), p(f(u),f(u))\}$



First-order Contexts: Examples

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

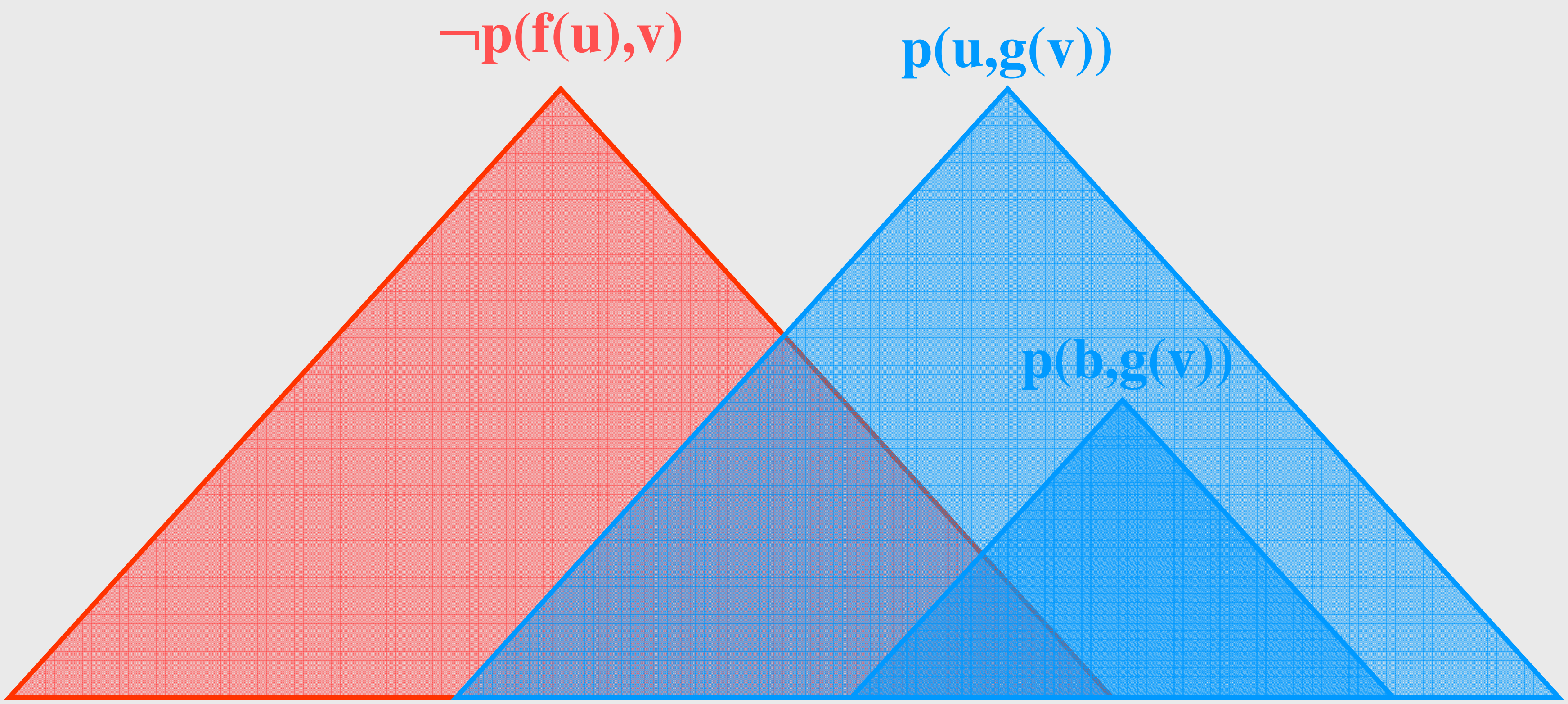
OK



First-order Contexts: Examples

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

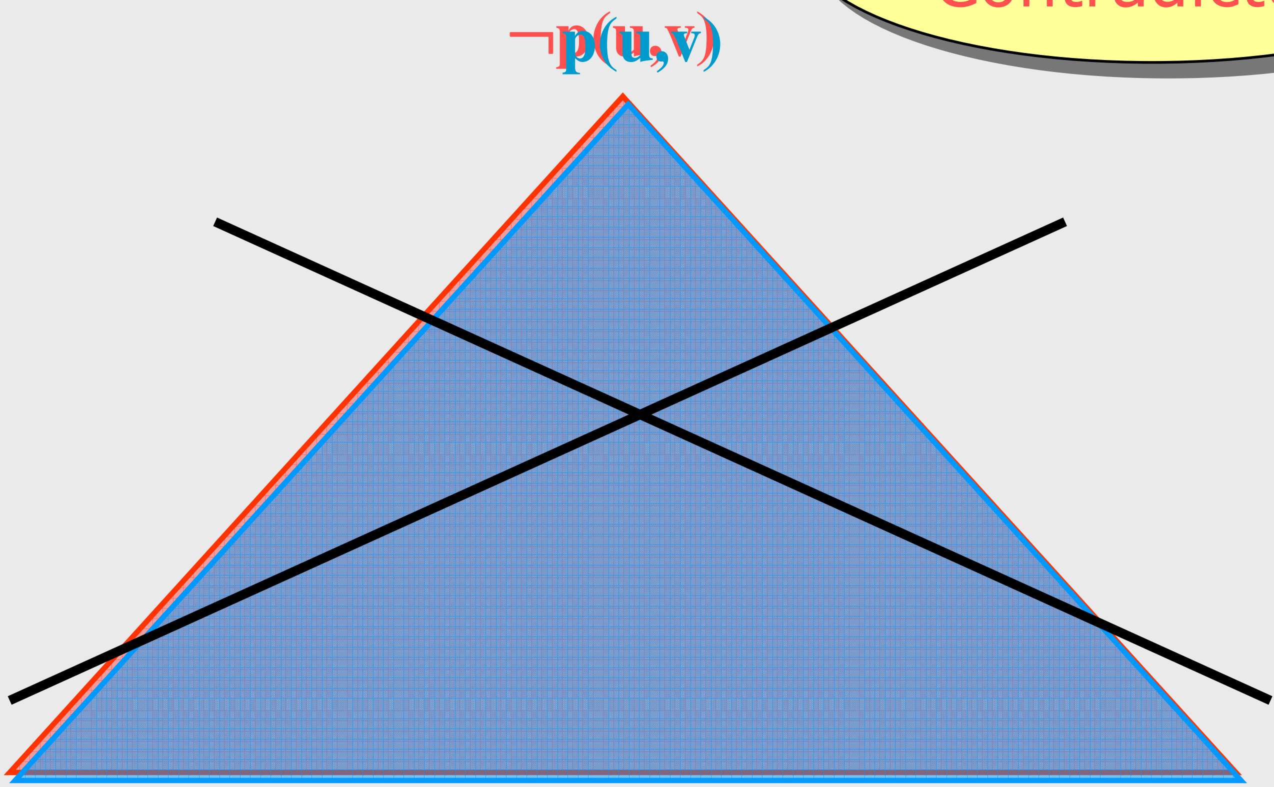
OK



First-order Contexts: Examples

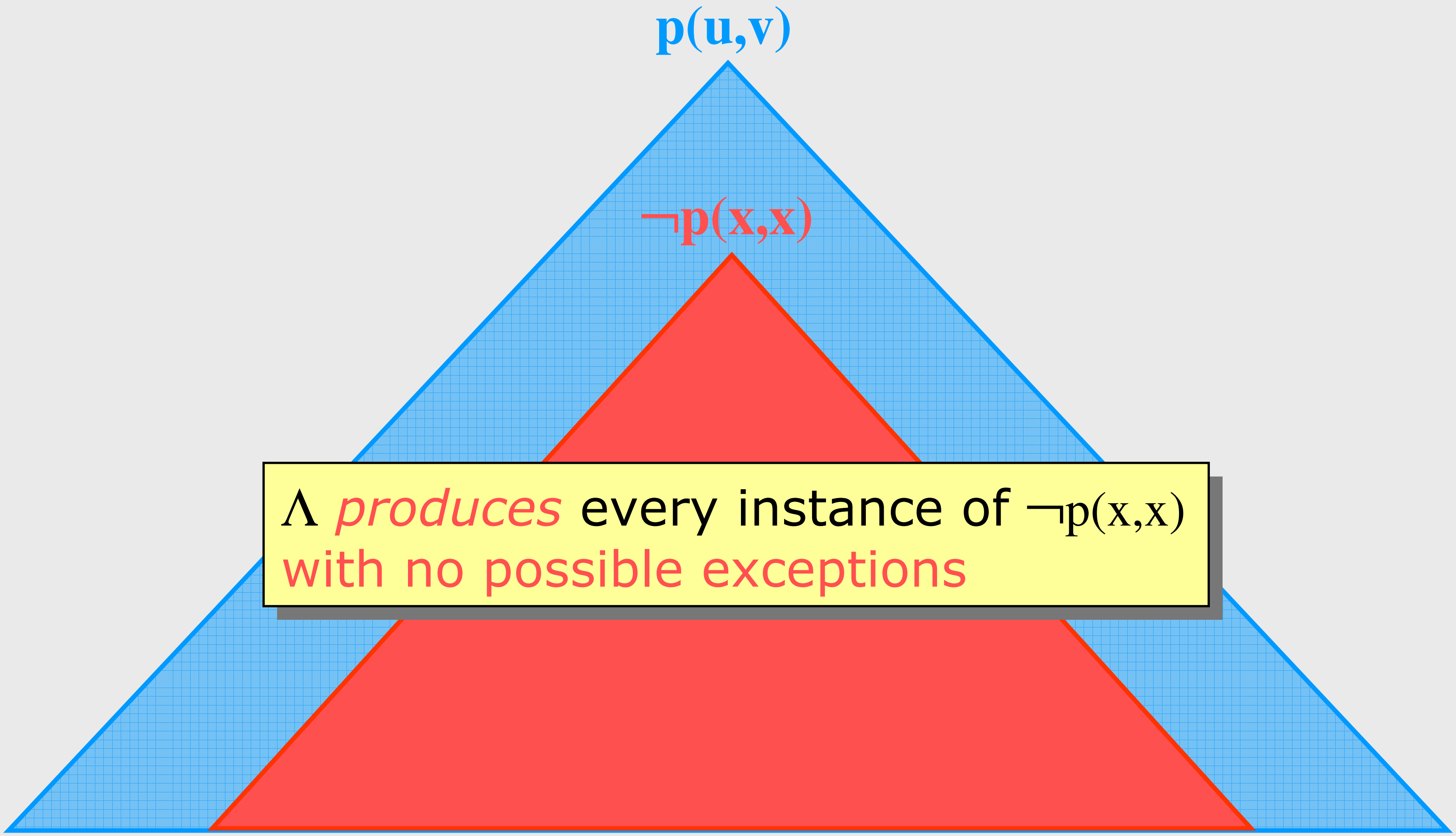
$\Lambda = \{\neg p(u,v),\ p(u,v)\}$

Not OK!
Contradictory



First-order Contexts: Examples

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

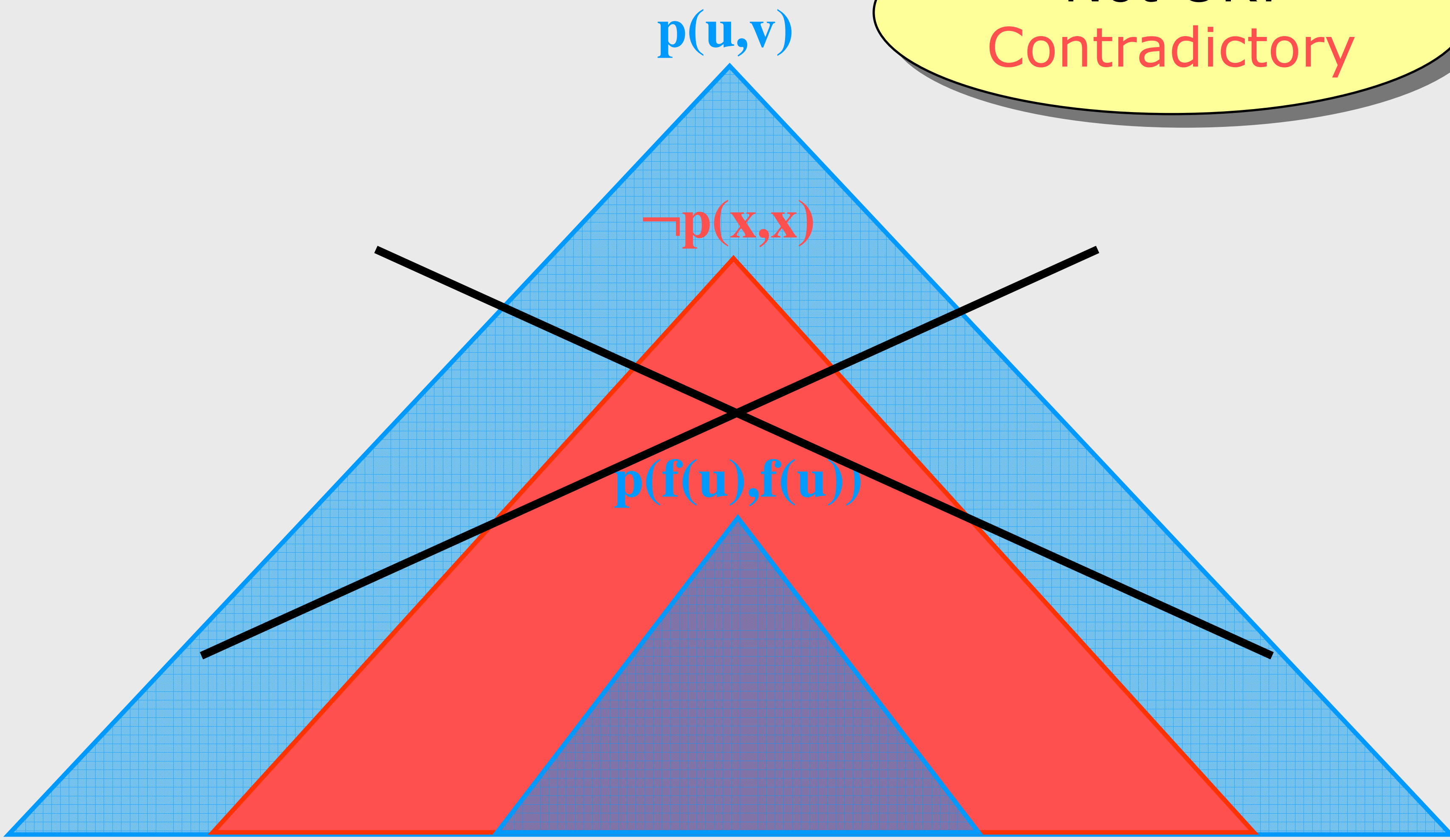


Λ *produces* every instance of $\neg p(x,x)$
with no possible exceptions

First-order Contexts: Examples

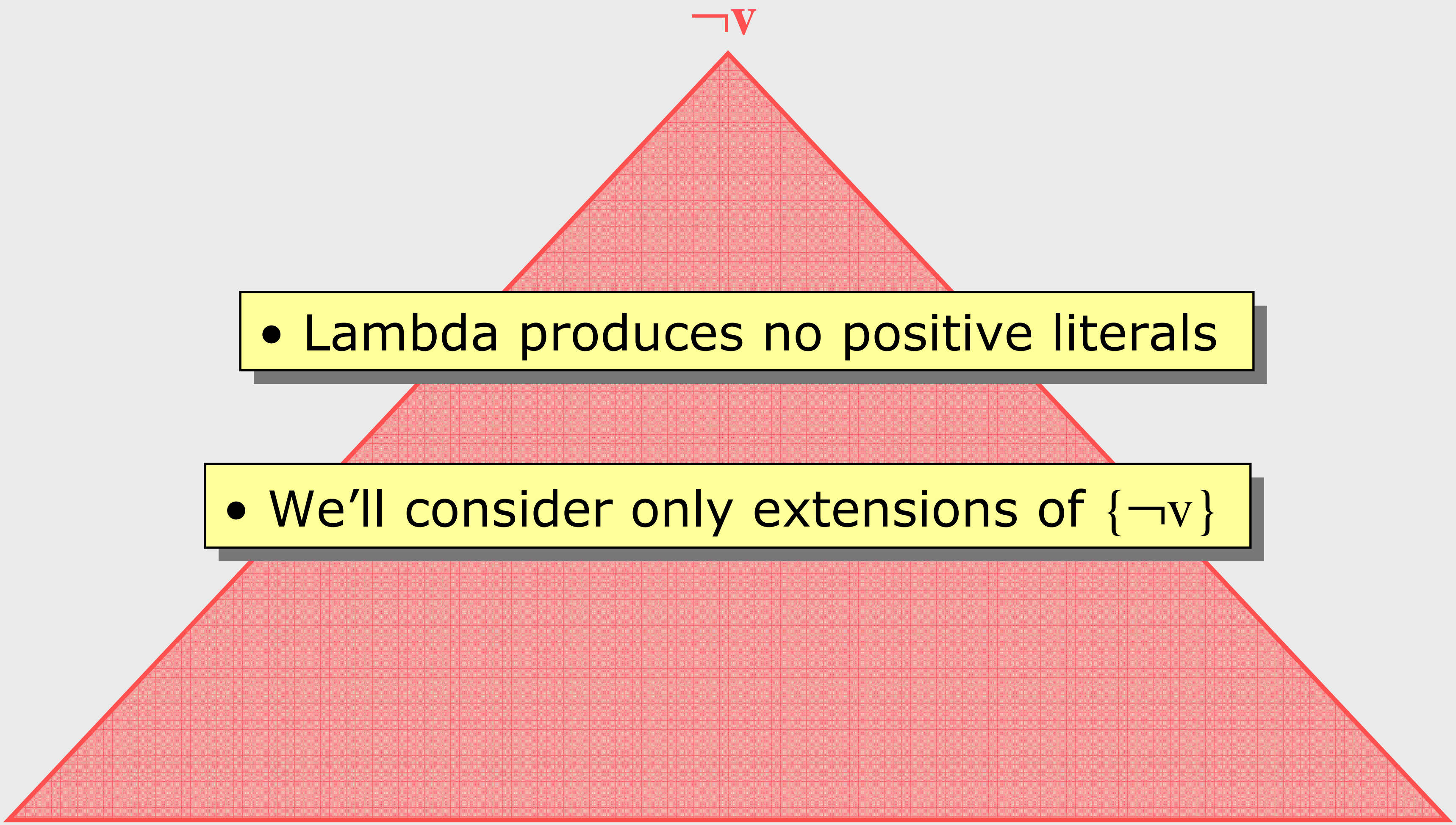
$\Lambda = \{p(u,v), \neg p(x,x), p(f(u),f(u)\}$

Not OK!
Contradictory



Initial Context

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



- Lambda produces no positive literals

- We'll consider only extensions of $\{\neg v\}$

Contexts and Interpretations

Let Λ be a non-contradictory context with parametric literals and universal literals

Λ denotes a Herbrand interpretation:

$$I_\Lambda = \left\{ L \;\middle|\; \begin{array}{l} L \text{ is ground and positive,} \\ L \text{ is produced by } \Lambda \end{array} \right\}$$

Checking $\models$

Let Λ be a non-contradictory context

Let $L_1 \vee \ldots \vee L_n$ be a (parameter-free) clause

If not $I_\Lambda \models L_1 \vee \ldots \vee L_n$ then

there are fresh variants $K_1, \ldots, K_n$ of literals in Λ

and a substitution σ

such that

σ is a simultaneous mgu of $\{K_1, \overline{L_1}\}, \ldots, \{K_n, \overline{L_n}\}$

- σ is called a *context unifier* (of the clause against Λ)

Checking $\Vdash$

Let Λ be a non-contradictory context

Let $L_1 \vee \ldots \vee L_n$ be a (parameter-free) clause

$I_\Lambda \Vdash \neg(L_1 \vee \ldots \vee L_n)$

iff the ... rals in Λ

and a ...

such t ...

1. σ ... $, \ldots, \{K_n, \overline{L_n}\}$
2. for each i, $Pars(K_i)\sigma \subseteq Pars$

Example

- $I_{\{\neg p(u,v)\}} \Vdash \neg(p(x,y) \vee p(x,x))$
- equivalently, match $\{p(x,y), p(x,x)\}$ against $\{p(\$,\$)\}$

The Model Evolution Calculus: Semantical View

$$(\textbf{assert})\quad \frac{\Lambda \quad \square \quad \Phi, L}{\Lambda, L \quad \square \quad \Phi, L} \quad \text{if } \begin{cases} \text{not } I_\Lambda \Vert\!= L \\ L \text{ not contr. with } \Lambda \end{cases}$$

$$(\textbf{subsume})\quad \frac{\Lambda \quad \square \quad \Phi, \; L \vee C}{\Lambda \quad \square \quad \Phi} \quad \text{if } I_\Lambda \Vert\!= L$$

Exactly the same as in DPLL!

$$(\textbf{resolve})\quad \frac{\Lambda \quad \square \quad \Phi, \; L \vee C}{\Lambda \quad \square \quad \Phi, \; C} \quad \text{if } I_\Lambda \Vert\!= \overline{L}$$

$$(\textbf{close})\quad \frac{\Lambda \quad \square \quad \Phi, \; C}{\Lambda \quad \square \quad \mathrm{W}} \quad \text{if } I_\Lambda \Vert\!= \neg C$$

The Model Evolution Calculus: Semantical View

$$(\textbf{split})\quad \frac{\Lambda \;\square\; \Phi, C \vee L}{\Lambda, L\sigma \;\square\; \Phi, C \vee L \qquad \Lambda, (\overline{L\sigma})^{\text{sko}} \;\square\; \Phi, C \vee L} \quad \text{if } (*)$$

$$(*) \quad = \quad \begin{cases} 1) & \sigma \text{ is a context unifier of } (C \vee L) \text{ against } \Lambda \\ 2) & \sigma \text{ is admissible} \\ 3) & \cancel{L\sigma \text{ is not mixed}} \quad \boxed{\text{No Longer}} \\ 4) & L\sigma \text{ not contr. with } \Lambda \\ 5) & (\overline{L\sigma})^{\text{sko}} \text{ not contr. with } \Lambda \end{cases}$$

The Split Rule: Example

First, identify falsified clause instance:

Clauses:

Now, split with

$\mathrm{abs}(x) \geq u \quad | \quad \neg \mathrm{abs}(c) \geq u$

Context: $\mathrm{abs}(x) \geq 0$ $\quad v \geq u \quad \neg v$

σ context unifier

$$\underbrace{\neg(\mathrm{abs}(x) \geq 0)}_{\text{Permanenly falsified}} \vee \underbrace{\neg(0 \geq u) \vee (\mathrm{abs}(x) \geq u)}_{\text{remainder}}$$

σ admissible context unifier

Example

$$
\begin{array}{rl}
\mathrm{abs}(u)\geq a & \underline{\mathrm{abs}(x)\geq a}, \ldots \\
& \Downarrow \ \textbf{assert} \\
\mathrm{abs}(x)\geq a,\ \underline{\mathrm{abs}(u)\geq a} & \mathrm{abs}(x)\geq a, \ldots \\
& \Downarrow \ \textbf{compact} \\
\mathrm{abs}(x)\geq a & \underline{\mathrm{abs}(x)\geq a}, \ldots \\
& \Downarrow \ \textbf{subsume} \\
\mathrm{abs}(x)\geq a & \underline{\neg\mathrm{abs}(f(x))\geq a} \vee p(x), \ldots \\
& \Downarrow \ \textbf{resolve} \\
\mathrm{abs}(x)\geq a & p(x), \ldots
\end{array}
$$

Further Notions

- Derivation tree
- Exhausted/closed branch
- Derivation/refutation
- Limit tree
- Fair limit tree/derivation

Main Results: Completeness

Let $(\Lambda_i \,\square\, \Phi_i)_{i<\kappa}$, with $\kappa \leq \omega$, be an exausted branch in a fair limit tree of Φ_0.

Let $\Lambda = \bigcup_{i<\kappa} \bigcap_{i\leq j<\kappa} \Lambda_j$ and $\Phi = \bigcup_{i<\kappa} \bigcap_{i\leq j<\kappa} \Phi_j$.

If $W \notin \Phi$ then $I_\Lambda \models \Phi_0$.

Main Results: Soundness and Completeness

A clause set Φ_0 is unsatisfiable

iff

it has a refutation.

Main Results: Proof Convergence

If a clause set Φ_0 is unsatisfiable then every fair derivation of Φ_0 extends to a refutation.

Making ME Efficient

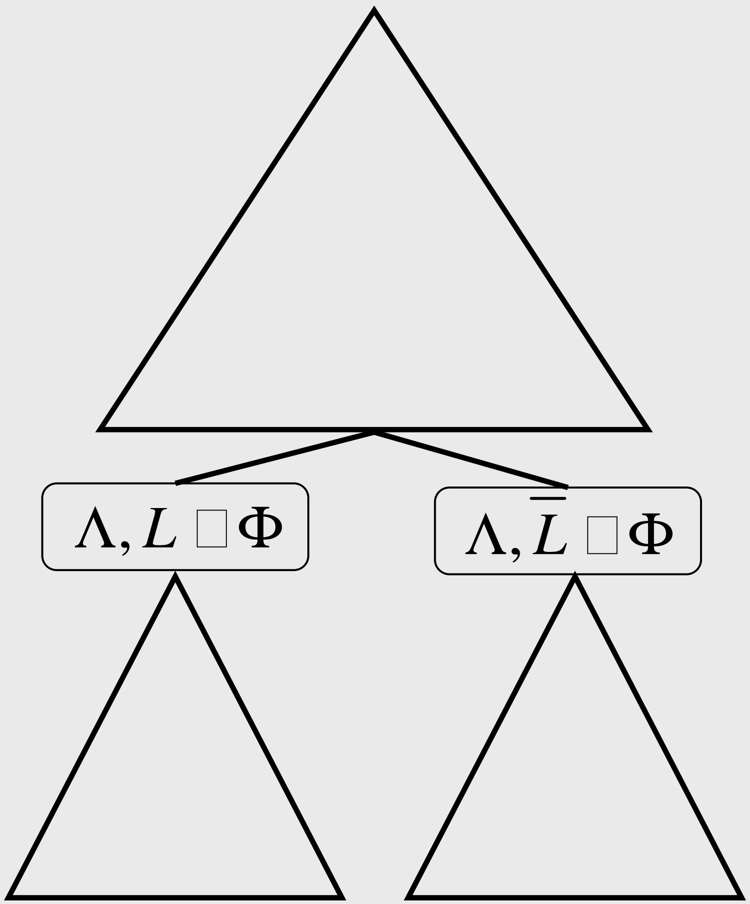
Well-known DPLL improvements:

- **Literal selection** strategies
  Model Elimination:
  can exploit don't care nondeterminism
  for remainer literal to split on

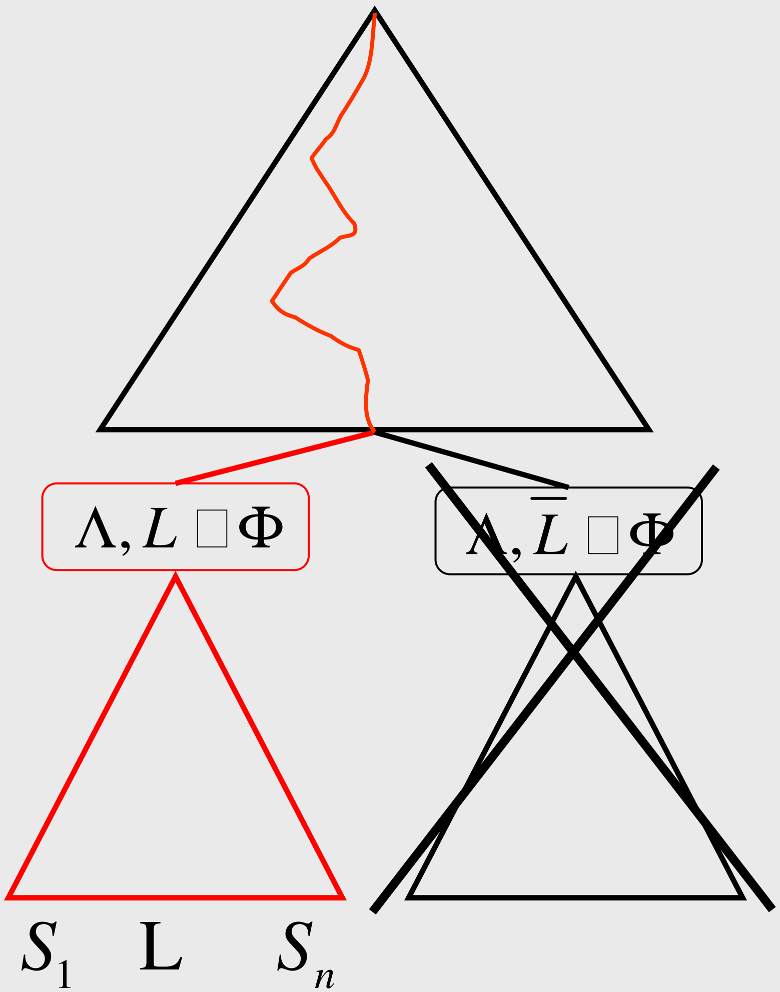
- **Learning** (lemma generation)
  not trivial – future work

- **Intelligent backtracking** (backjumping)

Backjumping



Backjumping



L not used to close left subtree

Conclusions

- **Full lifting of DPLL achieved**
- **Properties of DPLL preserved**
  - sound and complete
  - proof convergent
  - simplification rules
  - model generation paradigm
  - (no Commit rule as in FDPLL)
- **Abstract framework**
  - Wide range for fair strategies
  - Semantically justified redundancy criteria

Further Work

- Implement the calculus! (in progress)
- Lift DPLL optimizations (backjumping, lemma generation, …)
- Add equality
- Study decidable fragments
- Add nonmonotonic features
- Build-in theories
- …