# Proof Search in Conflict Resolution

## Lifting CDCL (Conflict-Driven Clause Learning) to First-Order Logic


Bruno Woltzenlogel Paleo

joint work with:

Daniyar Itegulov (ITMO, St. Petersburg, Russia)

Ezequiel Postan (National University of Rosario, Argentina)

John Slaney (Australian National University)

modus ponens

$$\frac{A \qquad A \rightarrow B}{B}$$

resolution

$$\frac{\Gamma_1 \rightarrow \Delta_1, A \qquad A', \Gamma_2 \rightarrow \Delta_2}{(\Gamma_1, \Delta_1 \rightarrow \Gamma_2, \Delta_2)\sigma}$$

hypothetical reasoning

$$\frac{\begin{matrix}[A] \\ \vdots \\ B\end{matrix}}{A \rightarrow B}$$

?

# Results of CASC (2016)

| Higher-order Theorems | Satallax 3.0 | Satallax 2.8 | LEO-II 1.7.0 | Leo+III 1.0 | Leo-III 1.0 | Isabelle 2015 |
|---|---|---|---|---|---|---|
| Solved/500 | 346/500 | 315/500 | 238/500 | 89/500 | 74/500 | 356/500 |
| Av. CPU Time | 22.10 | 19.45 | 20.93 | 48.37 | 42.79 | 81.08 |
| Solutions | 327/500 | 313/500 | 231/500 | 88/500 | 74/500 | 0/500 |

| Typed First-order Theorems +*-/ | Vampire 4.1 | VampireZ3 1.0 | CVC4 TFF-1.5.1 | Beagle 0.9.47 | Princess 160606 |
|---|---|---|---|---|---|
| Solved/500 | 419/500 | 380/500 | 343/500 | 300/500 | 342/500 |
| Av. CPU Time | 13.39 | 9.15 | 5.72 | 18.76 | 17.59 |
| Solutions | 419/500 | 380/500 | 343/500 | 300/500 | 271/500 |

| Typed First-order Non-theorems +*-/ | Beagle SAT-0.9.47 | CVC4 TFN-1.5.1 | Princess 160606 | CVC4 TFN-1.5 |
|---|---|---|---|---|
| Solved/50 | 10/50 | 9/50 | 8/50 | 8/50 |
| Av. CPU Time | [illegible].11 | 0.02 | 1.44 | 22.90 |

| First-order Theorems | Vampire 4.0 | Vampire 4.1 | E 2.0 | CVC4 FOF-1.5.1 | iProver 2.5 | leanCoP 2.2 | Prover9 1109a | Geo-III 2016C |
|---|---|---|---|---|---|---|---|---|
| Solved/500 | 457/500 | 447/500 | 392/500 | 329/500 | 278/500 | 168/500 | 101/500 | 54/500 |
| Av. CPU Time | 15.39 | 14.14 | 30.87 | 35.04 | 30.82 | 77.94 | 29.99 | 41.73 |
| Solutions | 453/500 | 447/500 | 392/500 | 328/500 | 274/500 | 168/500 | 98/500 | 54/500 |

| First-order Non-theorems | Vampire SAT-4.1 | Vampire SAT-4.0 | iProver SAT-2.5 | Nitpick 2015 | CVC4 FNT-1.5.1 | Geo-III 2016C | E FNT-2.0 | Refute 2015 |
|---|---|---|---|---|---|---|---|---|
| Solved/300 | 250/300 | 240/300 | 200/300 | 139/300 | 96/300 | 76/300 | 70/300 | 58/300 |
| Av. CPU Time | 40.11 | 6.45 | 30.28 | 37.86 | 22.43 | 13.69 | 16.31 | 69.09 |
| Solutions | 248/300 | 238/300 | 200/300 | 139/300 | 96/300 | 76/300 | 70/300 | 0/300 |

| Effectively Propositional CNF | iProver 2.5 | Vampire 4.1 | Vampire 4.0 | E 2.0 | Geo-III 2016C |
|---|---|---|---|---|---|
| Solved/300 | 229/300 | 222/300 | 222/300 | 101/300 | 10/300 |
| Av. CPU Time | 28.25 | 29.19 | 35.35 | 21.85 | 55.88 |

| Large Theory Batch Problems | Vampire LTB-4.0 | Vampire LTB-4.1 | Vampire LTB-4.1 | E LTB-2.0 | iProver LTB-2.5 | Prover9Pl 1.0 |
|---|---|---|---|---|---|---|
| Solved/600 | 403/600 | 398/600 | 396/600 | 305/600 | 298/600 | 85/200 |
| Av. WC Time | 11.62 | 9.54 | 8.05 | 12.56 | 35.07 | 14.41 |
| Solutions | 403/600 | 398/600 | 396/600 | 305/600 | 298/600 | 85/200 |

Very Sketchy Anatomy of Winning ATPs

First/Higher-Order Theorem Prover

# Let's Open the Black Box!

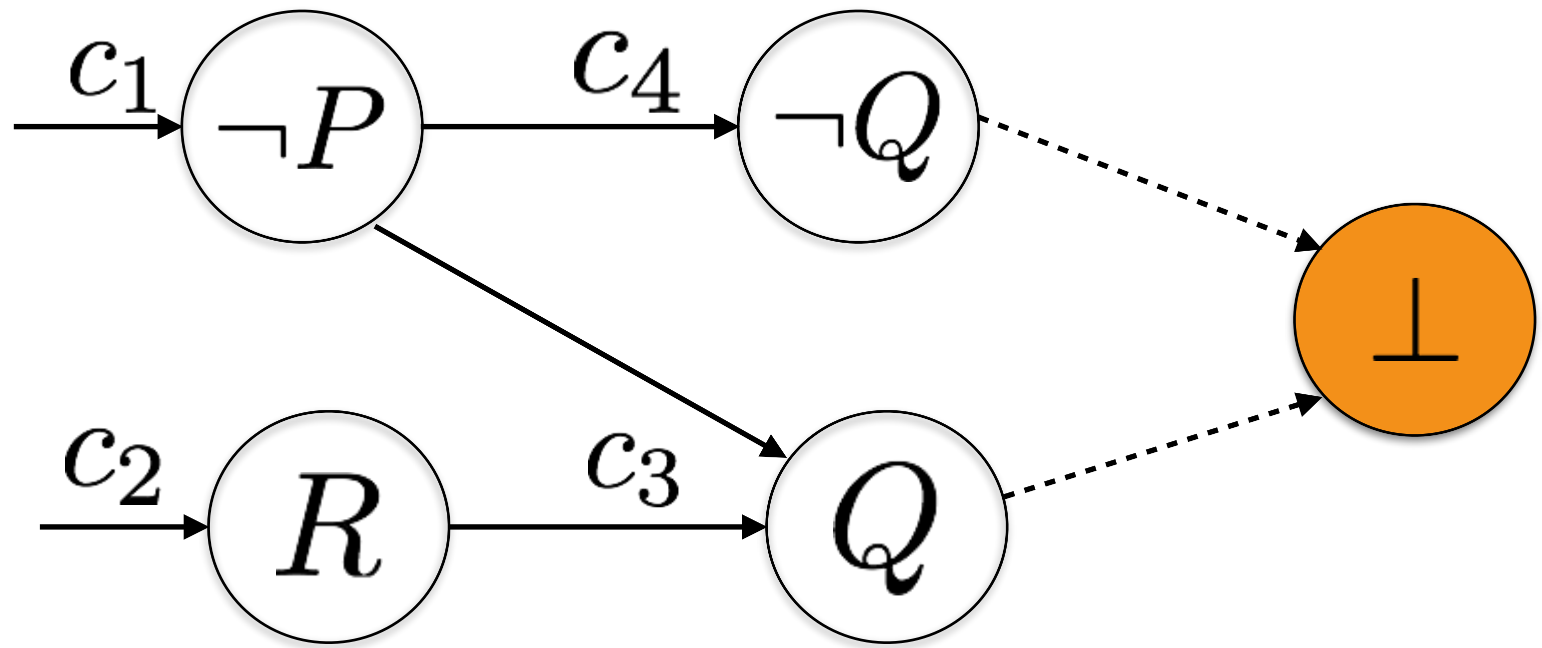
# Implication/Conflict Graphs: Unit Propagation

$c_1 : \neg P$

$c_2 : R$

$c_3 : \neg R \vee P \vee Q$

$c_4 : P \vee \neg Q$



# Unit-Propagating Resolution

$$\frac{\ell_1 \quad \dots \quad \ell_n \quad \bar{\ell}_1 \vee \dots \vee \bar{\ell}_n \vee \ell}{\ell}\ \mathbf{u}$$

$$\frac{\ell \quad \bar{\ell}}{\bot}\ \mathbf{c}$$

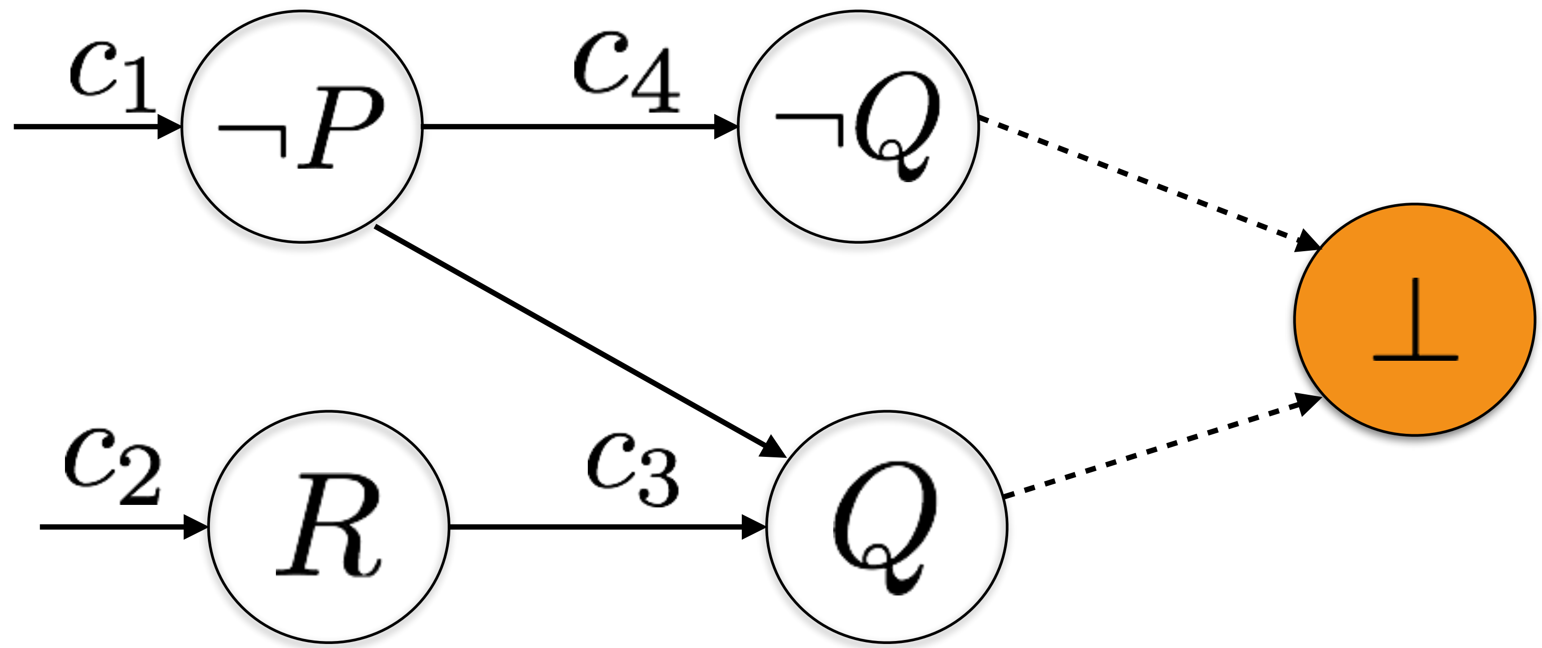
# Implication/Conflict Graphs: Unit Propagation

$c_1 : \neg P$

$c_2 : R$

$c_3 : \neg R \vee P \vee Q$

$c_4 : P \vee \neg Q$




$$\dfrac{\dfrac{c_2 : R \quad c_1 : \neg P \quad c_3 : \neg R \vee P \vee Q}{Q}\,\mathbf{u} \qquad \dfrac{c_1 \quad c_4 : P \vee \neg Q}{\neg Q}\,\mathbf{u}}{\bot}\,\mathbf{c}$$

# Implication/Conflict Graphs: Decision Literals

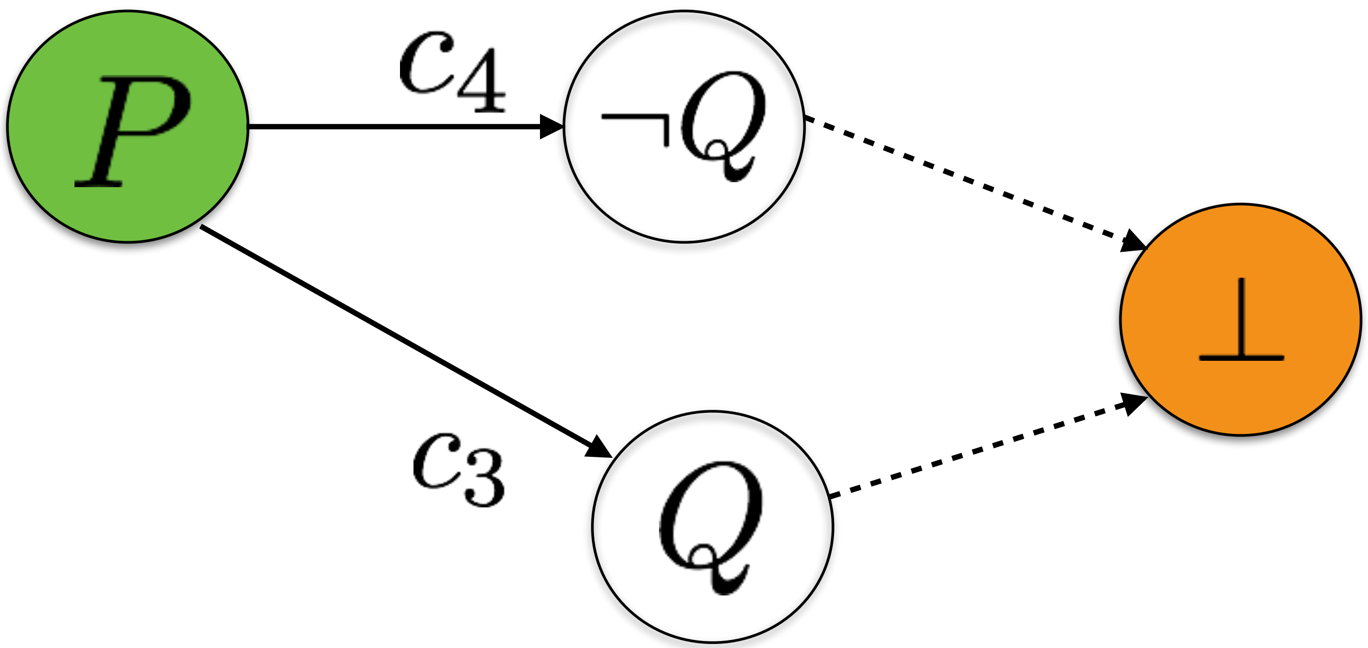
$$c_1 : P \vee Q$$

$$c_2 : P \vee \neg Q$$

$$c_3 : \neg P \vee Q$$

$$c_4 : \neg P \vee \neg Q$$

$$c_5 : \neg P$$



# Implication/Conflict Graphs

## Backtrack and Iterate…

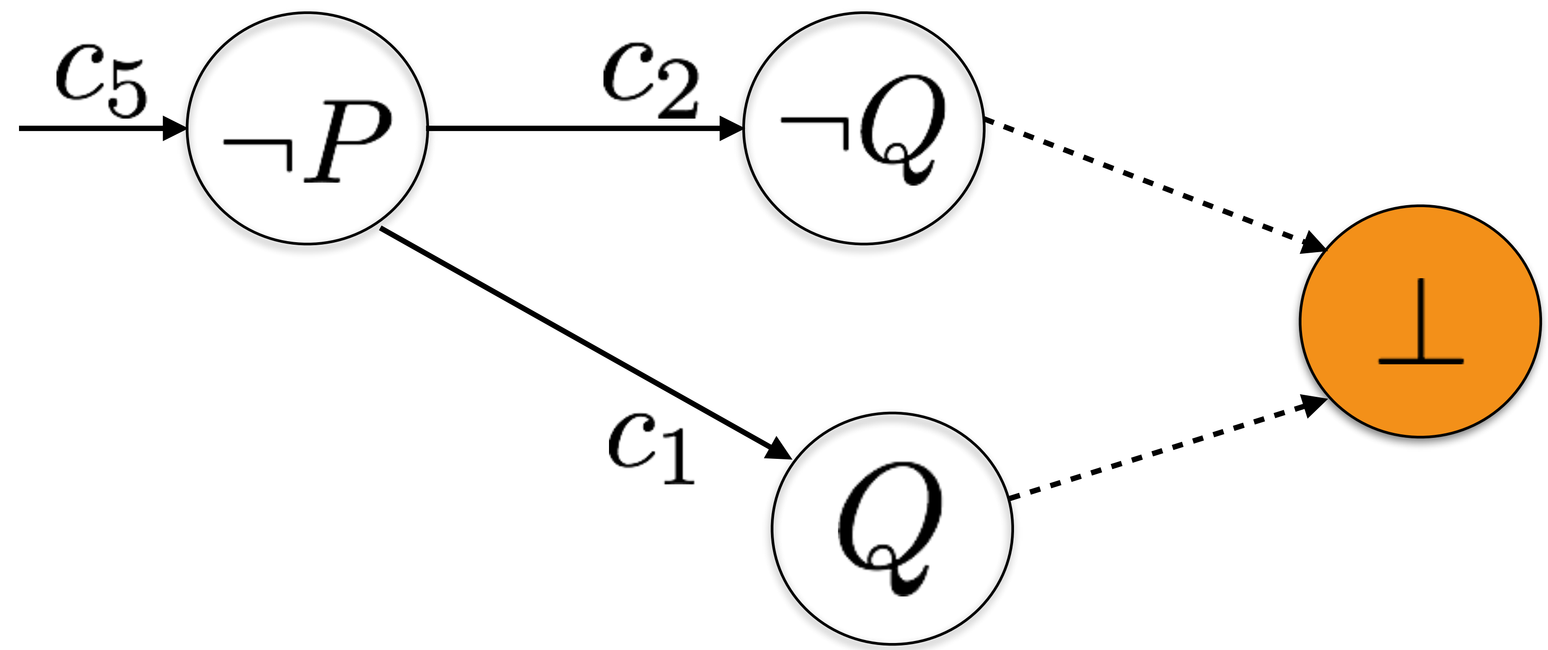
$c_1 : P \vee Q$

$c_2 : P \vee \neg Q$

$c_3 : \neg P \vee Q$

$c_4 : \neg P \vee \neg Q$

$c_5 : \neg P$



# Implication/Conflict Graphs: Decision Literals

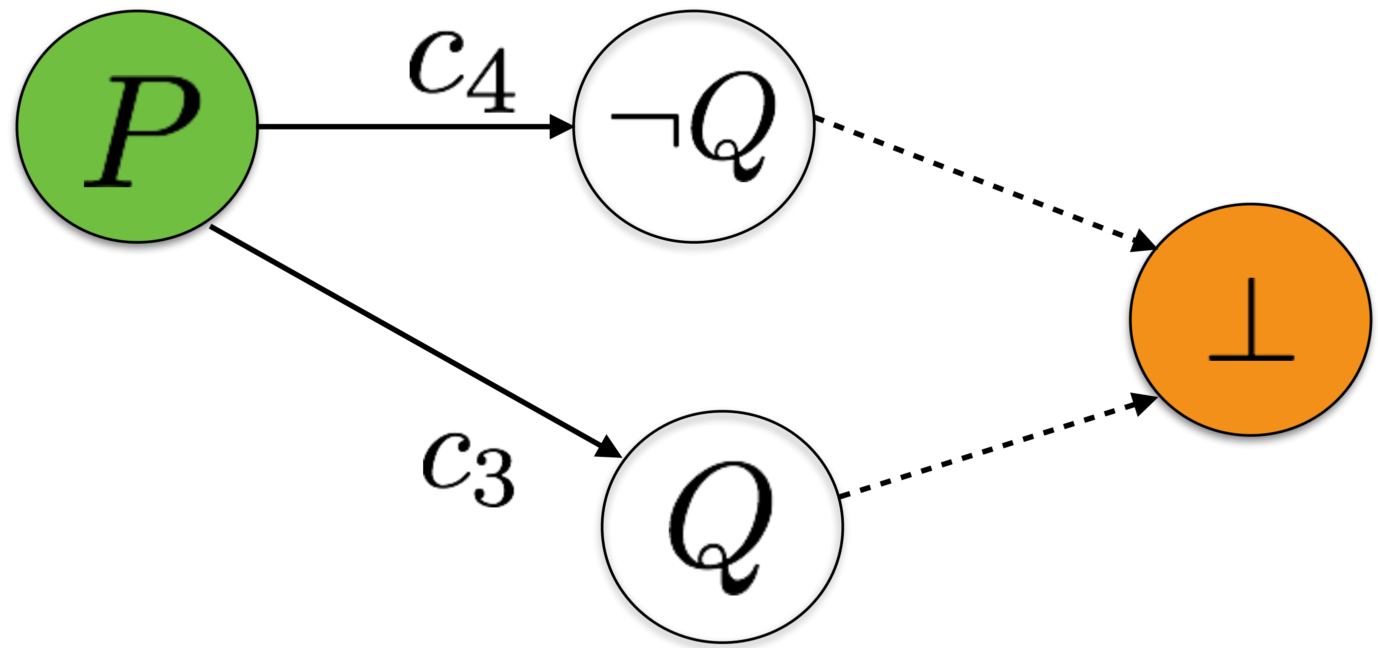
$c_1 : P \vee Q$

$c_2 : P \vee \neg Q$

$c_3 : \neg P \vee Q$

$c_4 : \neg P \vee \neg Q$

$c_5 : \neg P$




Decision literals behave like assumptions

learning a clause is like applying natural deduction's negation introduction rule

$$\dfrac{\begin{array}{c}[P] \\ \vdots \\ \bot\end{array}}{\neg P}\ \neg_I$$

# Decisions and Conflict-Driven Clause Learning

$$[\ell_1]^i \qquad\qquad [\ell_n]^i$$

$$\vdots \qquad\qquad\qquad \vdots$$

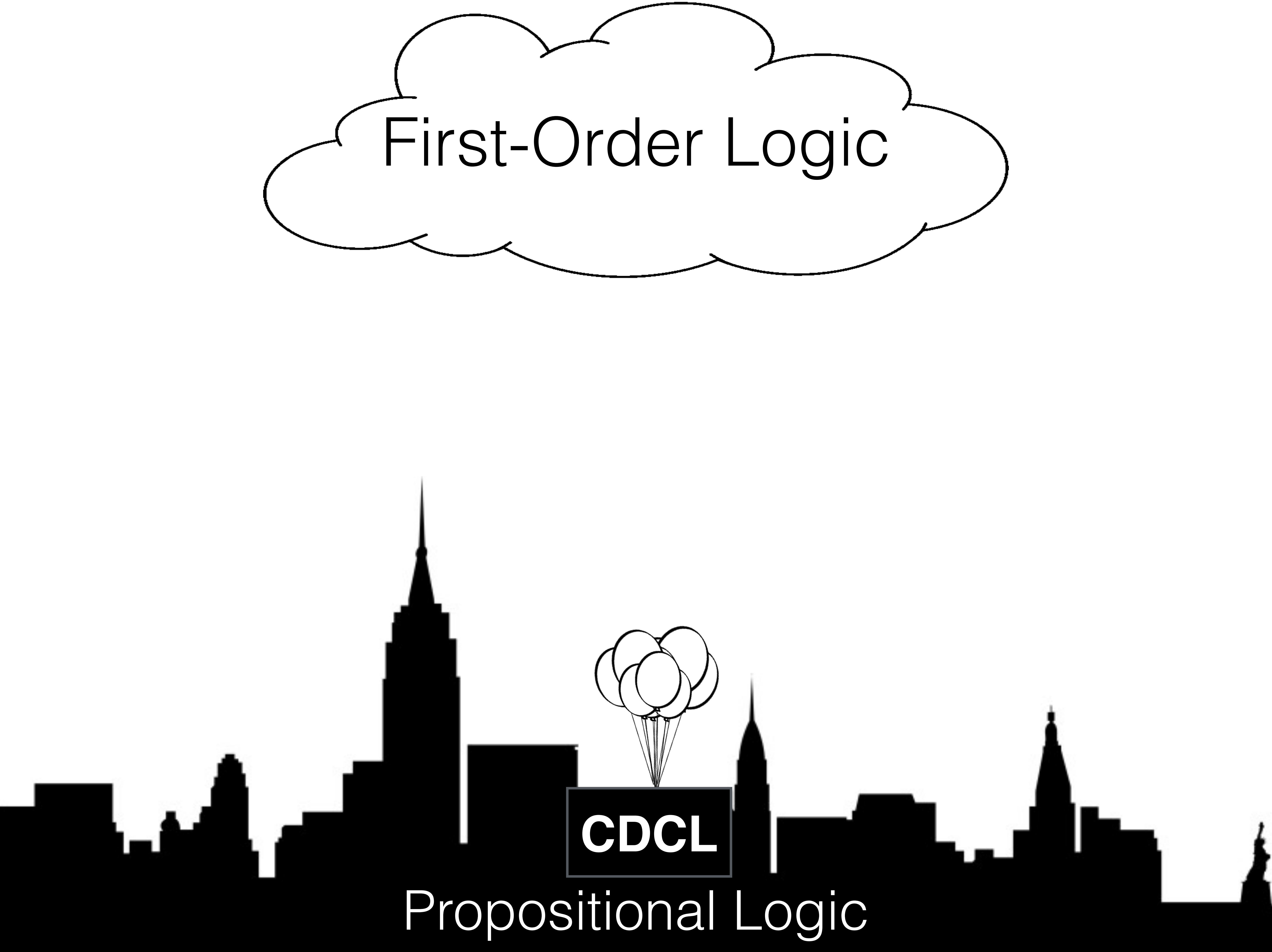
$$\vdots$$

$$\frac{\bot}{\bar{\ell_1} \vee \ldots \vee \bar{\ell_n}}\ \mathbf{cl}^i$$

This can also be a non-tree DAG

“cl” can be seen as a chain of negation/implication introductions

$$\neg P \equiv P \rightarrow \bot$$

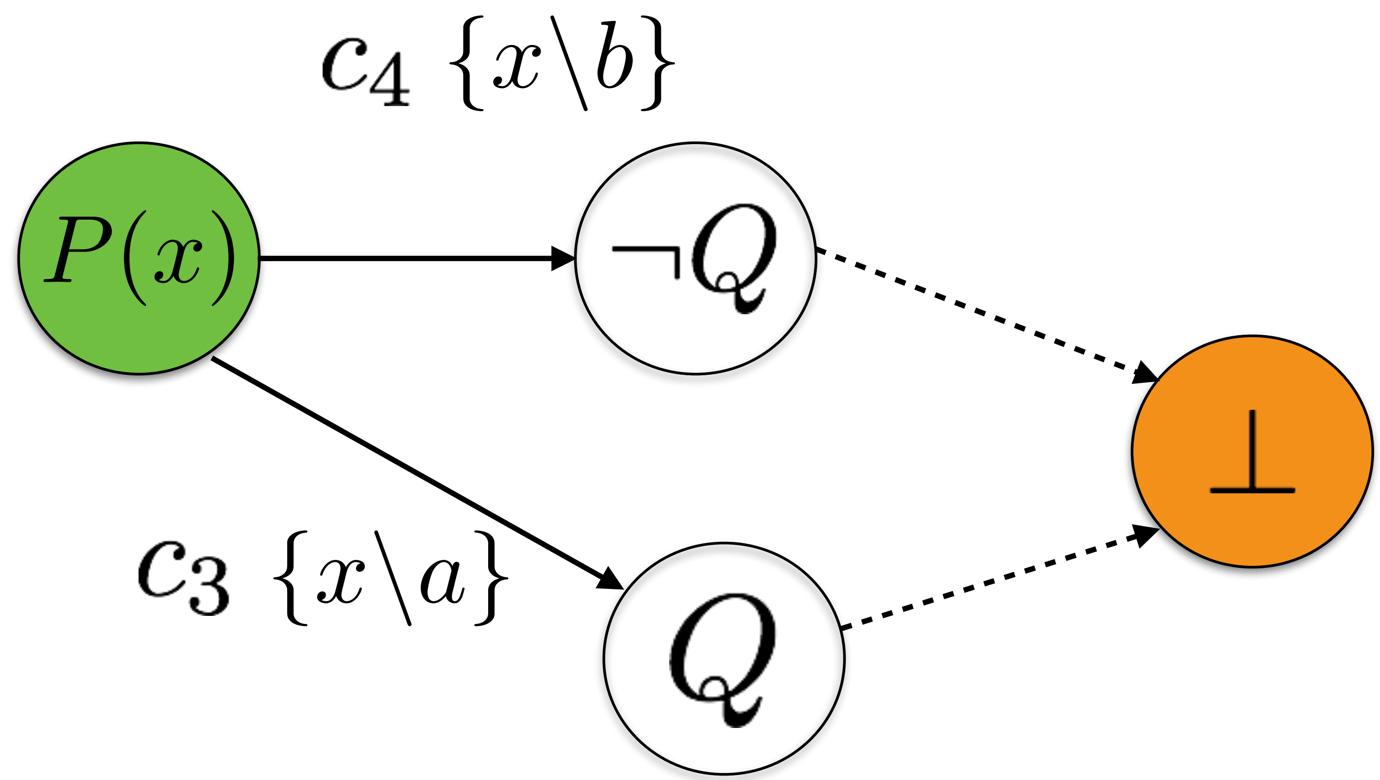

First-Order Logic
CDCL
Propositional Logic

# First-Order Unit-Propagation

$$\frac{\ell_1 \quad \ldots \quad \ell_n \quad \bar{\ell'_1} \vee \ldots \vee \bar{\ell'_n} \vee \ell}{\ell\ \sigma}\ \mathbf{u}(\sigma)$$

$$\frac{\ell \quad \bar{\ell'}}{\bot}\ \mathbf{c}(\sigma)$$

$c_1 : P(z) \vee Q$
$c_2 : P(y) \vee \neg Q$
$c_3 : \neg P(a) \vee Q$
$c_4 : \neg P(b) \vee \neg Q$




**Which clause should we learn?**

~~$c_5 : \neg P(x)$~~ **?**

$c_5 : \neg P(a) \vee \neg P(b)$

# First-Order Conflict-Driven Clause Learning

$$
\begin{array}{cc}
[\ell_1]^{\overset{i}{1}} & [\ell_n]^{\overset{i}{n}} \\
\vdots\ (\sigma^1_1, \ldots, \sigma^1_{m_1}) & \vdots\ (\sigma^n_1, \ldots, \sigma^n_{m_n})
\end{array}
$$

$$
\dfrac{\begin{array}{c}\vdots \\ \bot\end{array}}{(\bar{\ell_1}\sigma^1_1 \vee \ldots \vee \bar{\ell_1}\sigma^1_{m_1}) \vee \ldots \vee (\bar{\ell_n}\sigma^n_1 \vee \ldots \vee \bar{\ell_n}\sigma^n_{m_n})}\ \mathbf{cl}^i
$$

# Refutational Completeness

(by simulation of the resolution calculus)

$$\dfrac{\overset{\vdots\ \psi_1}{\ell_1 \vee \ldots \vee \ell_n \vee \ell} \qquad \overset{\vdots\ \psi_2}{\overline{\ell'} \vee \ell'_1 \vee \ldots \vee \ell'_m}}{(\ell_1 \vee \ldots \vee \ell_n \vee \ell'_1 \vee \ldots \vee \ell'_m)\ \sigma}\ \mathbf{r}(\sigma)$$

$$\dfrac{\dfrac{\dfrac{[\overline{\ell_1}]^1 \quad \ldots \quad [\overline{\ell_n}]^1 \quad \overset{\vdots\ \varphi_1}{\ell_1 \vee \ldots \vee \ell_n \vee \ell}}{\ell}\ \mathbf{u}(\varepsilon) \qquad \dfrac{[\overline{\ell'_1}]^1 \quad \ldots \quad [\overline{\ell'_m}]^1 \quad \overset{\vdots\ \varphi_2}{\overline{\ell'} \vee \ell'_1 \vee \ldots \vee \ell'_m}}{\overline{\ell'}}\ \mathbf{u}(\varepsilon)}{\bot}\ \mathbf{c}(\sigma)}{(\ell_1 \vee \ldots \vee \ell_n \vee \ell'_1 \vee \ldots \vee \ell'_m)\ \sigma}\ \mathbf{cl}^1$$

# Refutational Completeness

(by simulation of the resolution calculus)

$$
\cfrac{\begin{array}{c} \vdots\ \varphi' \\ \ell \vee \ell' \vee \ell_1 \vee \ldots \vee \ell_m \end{array}}{(\ell \vee \ell_1 \vee \ldots \vee \ell_m)\ \sigma}\ \mathbf{f}(\sigma)
$$

$$
\cfrac{\cfrac{\cfrac{\psi : [\overline{\ell}\sigma]^{1}_{1} \quad \psi \quad [\overline{\ell_1}]^{1}_{2} \quad \ldots \quad [\overline{\ell_{m-1}}]^{1}_{m} \quad \begin{array}{c} \vdots\ \varphi' \\ \ell \vee \ell' \vee \ell_1 \vee \ldots \vee \ell_m \end{array}}{\ell_m\ \sigma}\ \mathbf{u}(\sigma) \qquad [\overline{\ell_m}]^{1}_{m+1}}{\bot}\ \mathbf{c}(\sigma)}{(\ell \vee \ell_1 \vee \ldots \vee \ell_m)\ \sigma}\ \mathbf{cl}^{1}
$$

**The simulation is linear**

# Soundness

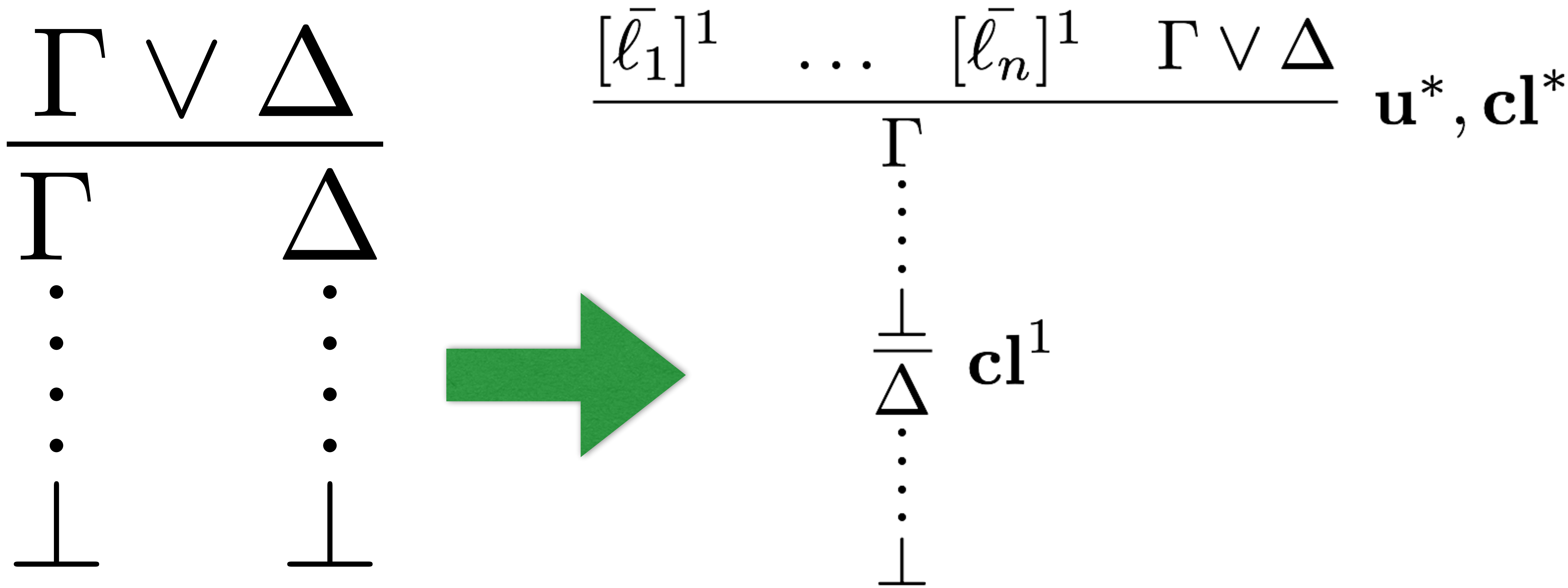
(via simulation by natural deduction)

Step 1:
ground the conflict resolution proof
(expand DAG to tree when necessary)

Step 2:
simulate each unit propagating resolution or conflict
by a chain of implication eliminations.
simulate each conflict driven clause learning inference
by a chain of negation/implication introductions.

**Conflict Resolution = "Chained" Natural Deduction with Unification**

# *A Side-Remark:* Linear Simulation of Splitting




Now we could even split when

$$var(\Gamma) \cap var(\Delta) \neq \emptyset$$

JAR Paper accepted in January 2017

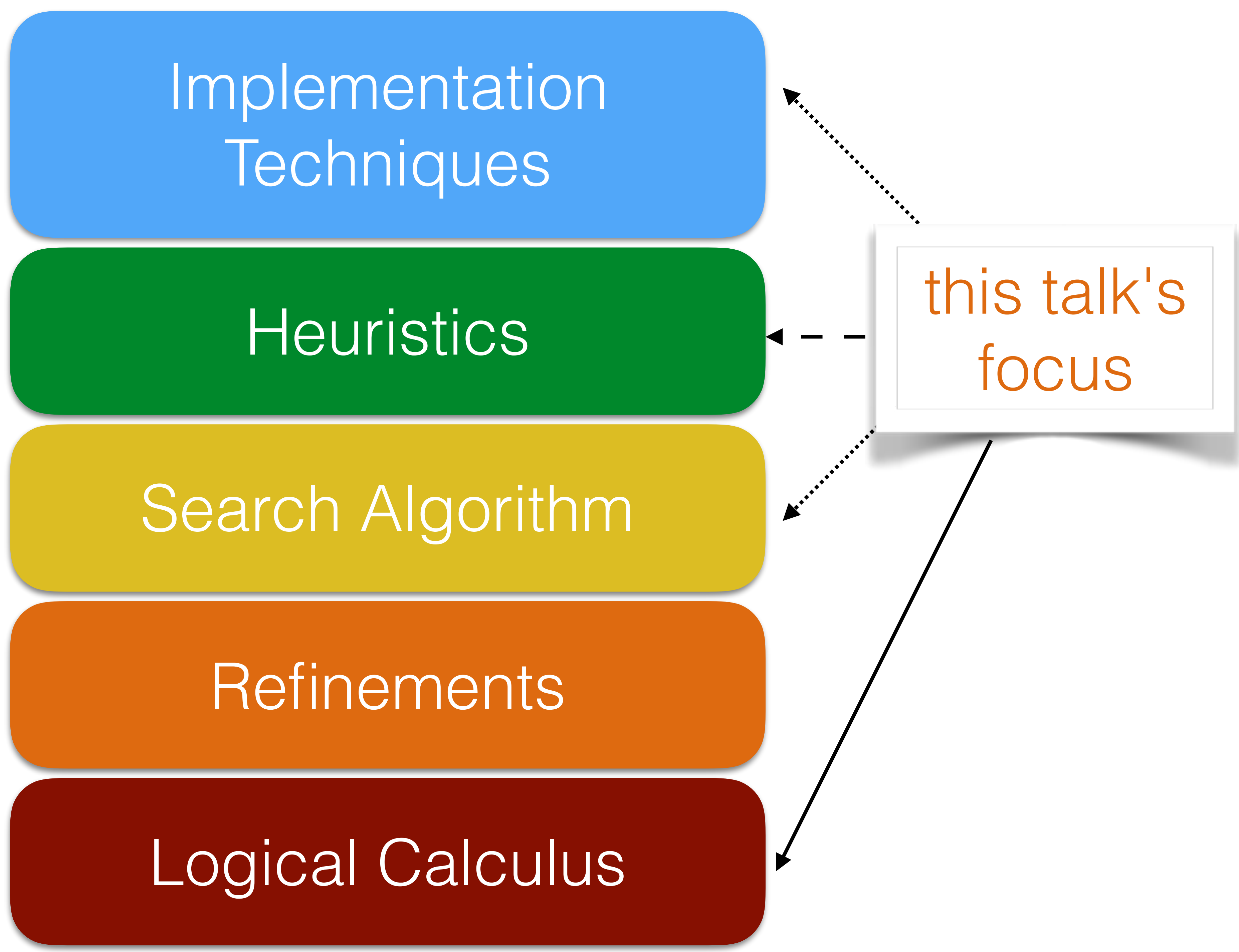
**Journal of Automated Reasoning manuscript No.**
(will be inserted by the editor)

# Conflict Resolution

## a First-Order Resolution Calculus with Decision Literals and Conflict-Driven Clause Learning

**John Slaney · Bruno Woltzenlogel Paleo**

# A Theorem Prover is much more than a Logical Calculus



# Pandora’s Box

4 "evils"
that attack
first-order
logic
but not
propositional
logic

# 1: Non-Termination of First-Order Unit Propagation

$$c_1 : P \vee Q$$

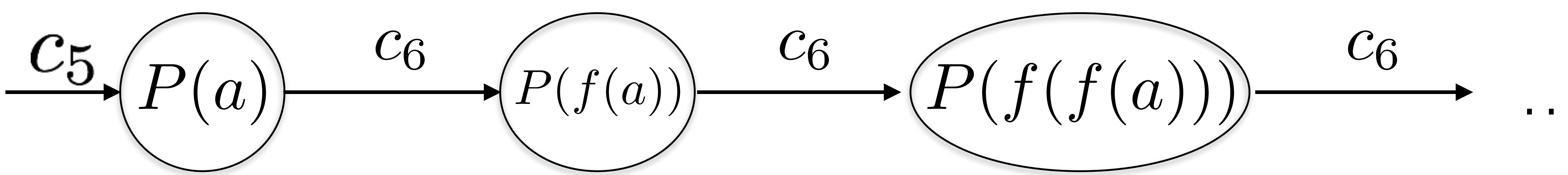
$$c_2 : P \vee \neg Q$$

$$c_3 : \neg P \vee Q$$

$$c_4 : \neg P \vee \neg Q$$

$$c_5 : P(a)$$

$$c_6 : \neg P(x) \vee P(f(x))$$




**Note:**

this problem will not occur in some decidable fragments (e.g. Bernays-Schönfinkel)

# Solutions

1) Ignore the non-termination.

2) Bound the propagation…

A) … by the depth of the propagation

B) … by the depth of terms occurring in propagated literals

and make decisions when the bound is reached, and then increase the bound.

# 2: Absence of Uniformly True Literals in Satisfied Clauses

$$\{p(X) \lor q(X), \neg p(a), p(b), q(a), \neg q(b)\}$$

is a satisfiable clause set

but there is no model where

$p(X)$ is uniformly true

or

$q(X)$ is uniformly true

This makes it harder to detect when
all clauses are already satisfied
(and, therefore, that we can stop the search)

# Solutions

1) Ignore the problem, and accept that
some satisfiable problems will not be solved.
(not so bad, if we focus on unsatisfiable problems)

2) Keep track of "useless decisions"
and consider a clause to be satisfied
when all its literals are useless decisions.

$$\{p(X) \vee q(X), \neg p(a), p(b), q(a), \neg q(b)\}$$

$p(X)$ and $q(X)$ are useless decisions

they lead to subsumed conflict-driven learned clauses

# 3: Propagation without Satisfaction

In a model containing $\neg p(a)$

The clause $p(X) \vee q(X)$ becomes propagating

and propagates $q(a)$ into the model

but having $q(a)$ in the model

does not make the clause satisfied

Even after propagation

a clause may be needed for other propagations

# Solution

1) Check whether the propagating clause became *uniformly satisfied*.

If so, then it won't be needed in future propagations

# 4: Quasi-Falsification without Propagation

In a model containing $\neg p(a)$ and $\neg q(b)$

the clause

$$p(X) \vee q(X) \vee r(X)$$

is quasi-falsified
(because its first two literals are false)

but $r(X)$ cannot be propagated

Moreover, detection of false literals needs
to take unification into account

This prevents direct lifting of
*two watched literals* data structure

# Solution

For each literal L occurring in a clause,
keep a hashset of literals in the model that are duals of instances of L.

If all literals of a clause except one have a non-empty hashset associated with it, the clause is quasi-falsified.

This allows quicker detection of quasi-falsified clauses in a manner that resembles two-watched literals

The set of quasi-falsified clauses is an over-approximation of the set of clauses that can propagate

# Implementation

# The Scavenger 0.1 Theorem Prover

Implemented in Scala

by me and two Google Summer of Code students:
Daniyar Itegulov and Ezequiel Postan

Open-Source: http://gitlab.com/aossie/Scavenger

**GSoC stipends available this year again!**

Google Summer of Code

www.aossie.org

Deadline: 3 April

AOSSIE
AUSTRALIAN OPEN SOURCE SOFTWARE INNOVATION AND EDUCATION

# Basic Data Structures

terms and formulas are simply-typed lambda expressions

*future work:*
*extend Conflict Resolution and Scavenger*
*to higher-order logic*

clauses are immutable sequents
(antecedent and succedent are sets)

# Proofs are DAGs of Proof Nodes

```
abstract class CRProofNode extends ProofNode[Clause, CRProofNode] {
  def findDecisions(sub: Substitution): Clause = {
    this match {
      case Decision(literal) =>
        !sub(literal)
      case conflict @ Conflict(left, right) =>
        left.findDecisions(conflict.leftMgu) union right.findDecisions(conflict.rightMgu)
      case UnitPropagationResolution(left, right, _, leftMgus, _) =>
        // We don't need to traverse right premise, because it's either initial clause or conflict driven clause
        left
          .zip(leftMgus)
          .map {
            case (node, mgu) => node.findDecisions(mgu(sub))
          }
          .fold(Clause.empty)(_ union _)
      case _ =>
        Clause.empty
    }
  }
```

# each inference rule is a small class

```scala
class Axiom(override val conclusion: Clause) extends CRProofNode {
  def auxFormulasMap = Map()
  def premises       = Seq()
}

case class Decision(literal: Literal) extends CRProofNode {
  override def conclusion: Clause = literal.toClause
  override def premises: Seq[CRProofNode] = Seq.empty
}

case class ConflictDrivenClauseLearning(conflict: Conflict) extends CRProofNode {
  val conflictDrivenClause = conflict.findDecisions(Substitution.empty)
  override def conclusion: Clause = conflictDrivenClause
  override def premises: Seq[CRProofNode] = Seq(conflict)
}
```

# each inference rule is a small class

```scala
case class UnitPropagationResolution private (left: Seq[CRProofNode], right: CRProofNode,
    desired: Literal, leftMgus: Seq[Substitution], rightMgu: Substitution) extends CRProofNode {
  require(left.forall(_.conclusion.width == 1), "All left conclusions should be unit")
  require(left.size + 1 == right.conclusion.width,
          "There should be enough left premises to derive desired")

  override def conclusion: Clause = desired

  override def premises: Seq[CRProofNode] = left :+ right
}
```

```scala
case class Conflict(leftPremise: CRProofNode, rightPremise: CRProofNode)
  extends CRProofNode {
  require(leftPremise.conclusion.width == 1, "Left premise should be a unit clause")
  require(rightPremise.conclusion.width == 1, "Right premise should be a unit clause")

  private val leftAux = leftPremise.conclusion.literals.head.unit
  private val rightAux = rightPremise.conclusion.literals.head.unit

  val (Seq(leftMgu), rightMgu) = unifyWithRename(Seq(leftAux), Seq(rightAux)) match {
    case None => throw new Exception("Conflict: given premise clauses are not resolvable")
    case Some(u) => u
  }

  override def premises = Seq(leftPremise, rightPremise)
  override def conclusion: Clause = Clause.empty
}
```

# Main Search Loop: 3 variants

1. EP-Scavenger: ignore non-termination of unit-propagation (168 lines)

2. PD-Scavenger: bound propagation by propagation depth (342 lines)

3. TD-Scavenger: bound propagation by term depth (176 lines)

# Important Missing Features

(Urgent Future Work)

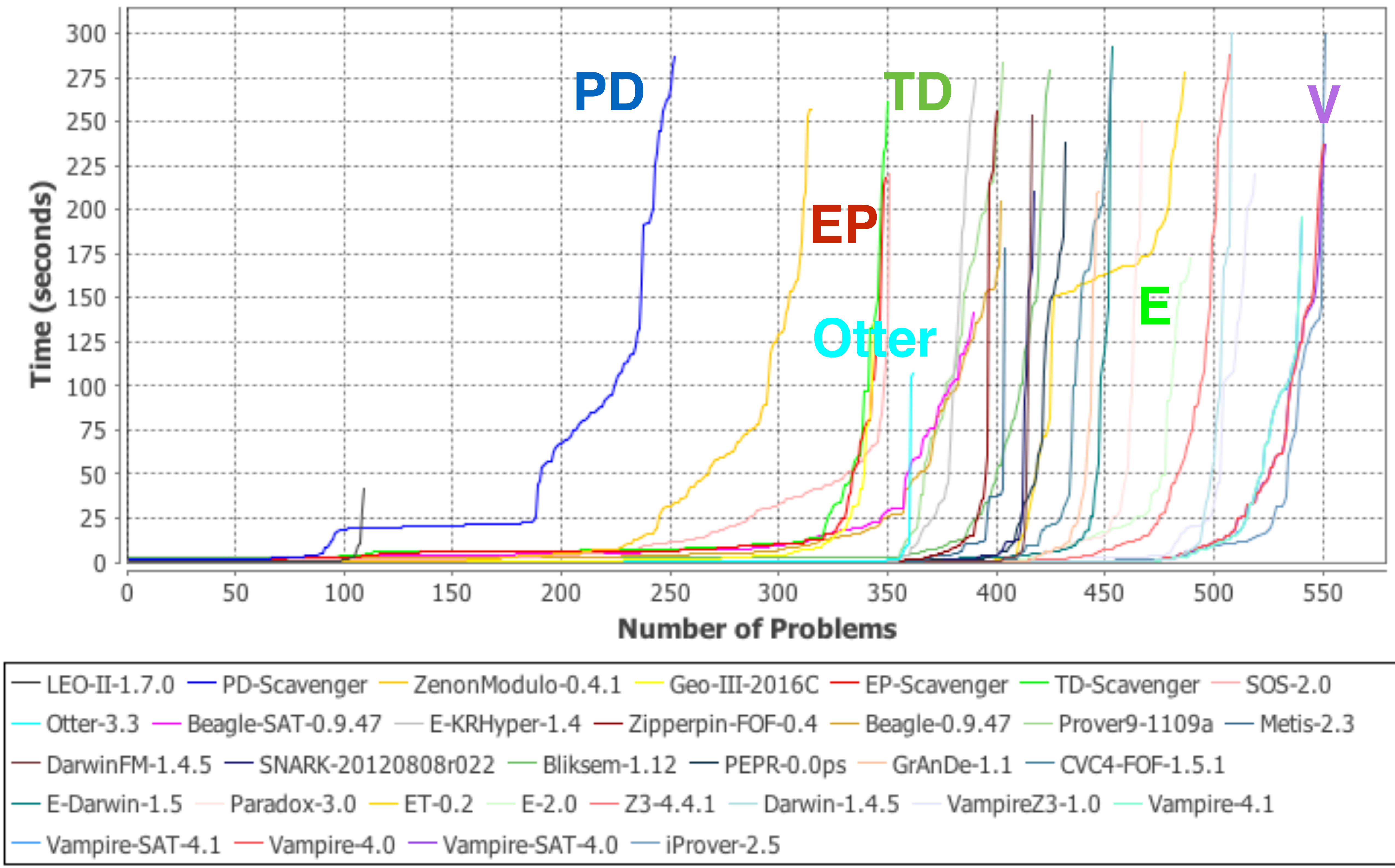
proper backtracking:

*Scavenger currently restarts and throws the model away after every conflict*

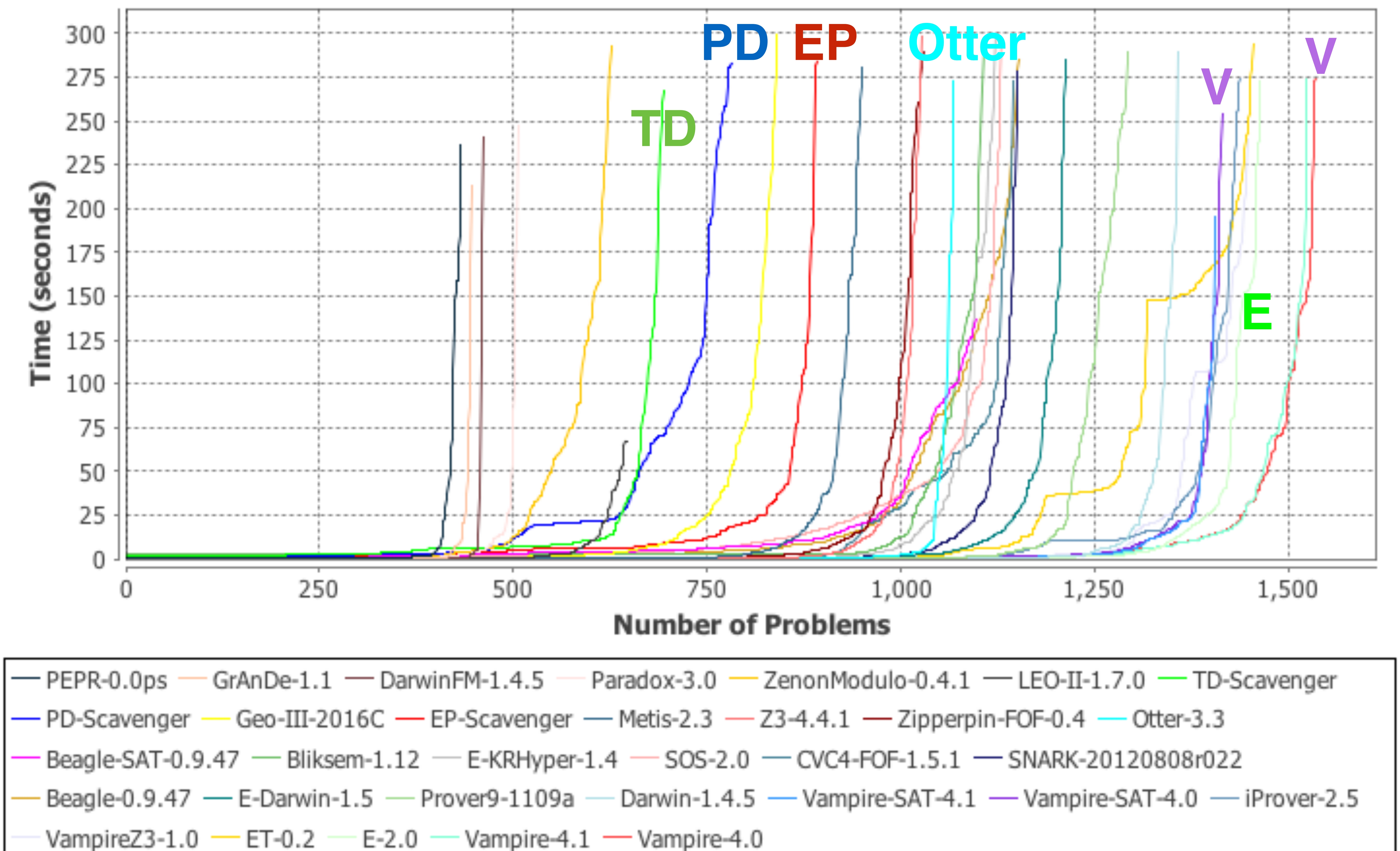
decision literal selection heuristics:

*Scavenger currently selects the first literal from a randomised queue*

# Preliminary Experiments




TPTP Unsat EPR CNF problems without Equality




TPTP Unsat CNF problems without Equality

# What about AI/ML?

| CDCL and CR | | Reinforcement Learning |
|---|---|---|
| current model | ↔ | state |
| selection of decision literals | ↔ | actions |
| learned clause | ↔ | punishment for (set of) bad decisions |
| heuristics selecting decision literals with highest scores | ↔ | policy selecting actions with highest values |

⋮

# Conclusions

modus ponens

$$\frac{A \qquad A \to B}{B}$$

resolution

$$\frac{\Gamma_1 \to \Delta_1, A \qquad A', \Gamma_2 \to \Delta_2}{(\Gamma_1, \Delta_1 \to \Gamma_2, \Delta_2)\sigma}$$

unit-resulting resolution

$$\frac{\ell_1 \qquad \dots \qquad \ell_n \qquad \ell_1 \to \dots \to \ell_n \to \ell}{\ell\sigma}\ \mathbf{u}(\sigma)$$

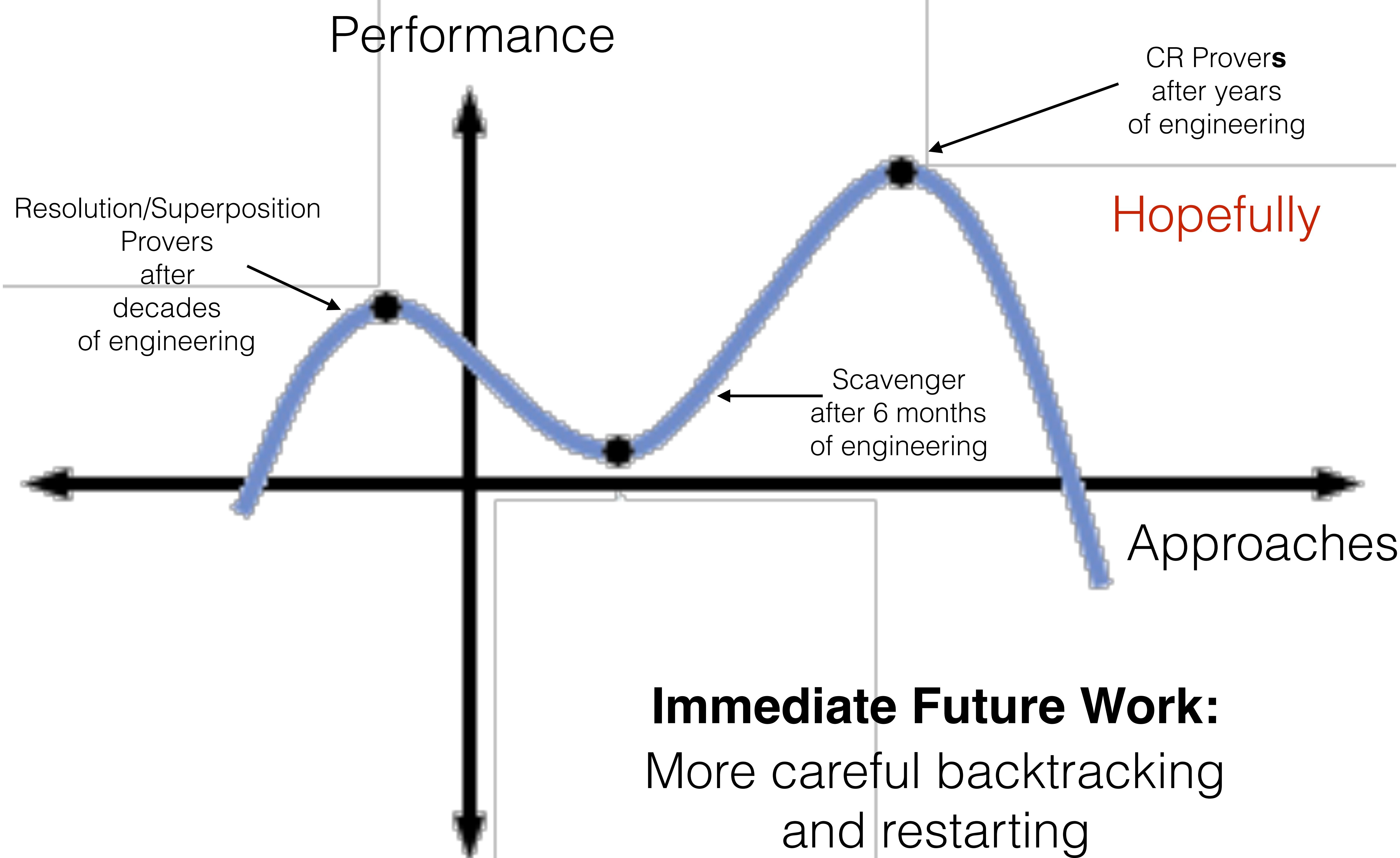
hypothetical reasoning

$$\frac{\begin{array}{c}[A] \\ \vdots \\ B\end{array}}{A \to B}$$

first-order CDCL

$$\frac{\begin{array}{c}[\ell_1]^{i}_{1} \qquad\qquad\qquad\qquad [\ell_n]^{i}_{n} \\ \vdots\ (\sigma^1_1, \dots, \sigma^1_{m_1}) \qquad \vdots\ (\sigma^n_1, \dots, \sigma^n_{m_n}) \\ \vdots \\ \bot\end{array}}{\ell_1\sigma^1_1, \dots, \ell_1\sigma^1_{m_1}, \dots, \ell_n\sigma^n_1, \dots, \ell_n\sigma^n_{m_n} \to \bot}\ \mathbf{cl}^i$$

?




**Immediate Future Work:**

More careful backtracking and restarting

Thank you!