Darwin - A Theorem Prover for the Model Evolution Calculus

IJCAR 2004 Workshop on Empirically Successful First Order Reasoning (Cork, Ireland, June 2004)

Peter Baumgartner
Max-Planck-Institut für Informatik
Saarbrücken, Germany

Alexander Fuchs
Universität Koblenz-Landau
Koblenz, Germany

Cesare Tinelli
The University of Iowa
Iowa City, IA 52242, USA

Background

- **The best modern SAT solvers (MiniSat, zChaff, BerkMin, ...) are based on DPLL.**
- **The Model Evolution Calculus (ME) is a direct lifting of DPLL to the first-order level.**
- **Darwin is the first implementation of ME.**

Overview

- **The Model Evolution Calculus in brief**
- **The Proof Procedure**
- **Implementation**
- **Evaluation**

DPLL - A Model Generation View

- **DPLL as a sequent-style calculus:**

$$\frac{\Lambda \vdash \Phi}{\Lambda' \vdash \Phi'} \quad \textbf{where} \quad \begin{cases} \Lambda \text{ literal set} \\ \Phi \text{ problem clause set} \end{cases}$$

- **The context Λ represents the currently assumed model for Φ, all not explicitely listed atoms are assumed to be false.**

- **If the context does not satisfy Φ, "repair" it by adding literals to Λ.**

DPLL - Example

Init. $\leadsto \{\}$ $\vdash \{p \lor q, q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s, p\}$

DPLL - Example

Init. $\leadsto \{\}$ $\quad \vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$

DPLL - Example

Init. $\leadsto \{\}$ $\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$

Assert p $\leadsto \{p\}$ $\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$

DPLL - Example

Init. $\rightsquigarrow \{\}$ $\quad\vdash \{p \lor q, q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s, p\}$

Assert p $\rightsquigarrow \{p\}$ $\quad\vdash \{p \lor q, q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s, p\}$

DPLL - Example

Init. $\leadsto \{\}$ $\quad \vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$

Assert p $\leadsto \{p\}$ $\quad \vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$

Subsume p $\leadsto \{p\}$ $\quad \vdash \{q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s\}$

DPLL - Example

Init. $\leadsto \{\}$ $\quad\vdash \{p \lor q, q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s, p\}$

Assert p $\leadsto \{p\}$ $\quad\vdash \{p \lor q, q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s, p\}$

Subsume p $\leadsto \{p\}$ $\quad\vdash \{q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s\}$

DPLL - Example

Init. $\rightsquigarrow \{\}$ $\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$

Assert p $\rightsquigarrow \{p\}$ $\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$

Subsume p $\rightsquigarrow \{p\}$ $\vdash \{q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s\}$

Resolve p $\rightsquigarrow \{p\}$ $\vdash \{q \vee \neg r \vee s, \neg q, \neg r \vee \neg s\}$

DPLL - Example

Init. $\leadsto \{\}$ $\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$

Assert p $\leadsto \{p\}$ $\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$

Subsume p $\leadsto \{p\}$ $\vdash \{q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s\}$

Resolve p $\leadsto \{p\}$ $\vdash \{q \vee \neg r \vee s, \neg q, \neg r \vee \neg s\}$

DPLL - Example

Init. $\rightsquigarrow \{\}$ $\quad\vdash \{p \lor q, q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s, p\}$

Assert p $\rightsquigarrow \{p\}$ $\quad\vdash \{p \lor q, q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s, p\}$

Subsume p $\rightsquigarrow \{p\}$ $\quad\vdash \{q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s\}$

Resolve p $\rightsquigarrow \{p\}$ $\quad\vdash \{q \lor \neg r \lor s, \neg q, \neg r \lor \neg s\}$

Assert ¬q $\rightsquigarrow \{p, \neg q\}$ $\quad\vdash \{\neg r \lor s, \neg r \lor \neg s\}$

DPLL - Example

Init. $\rightsquigarrow \{\}$ $\quad\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$

Assert p $\rightsquigarrow \{p\}$ $\quad\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$

Subsume p $\rightsquigarrow \{p\}$ $\quad\vdash \{q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s\}$

Resolve p $\rightsquigarrow \{p\}$ $\quad\vdash \{q \vee \neg r \vee s, \neg q, \neg r \vee \neg s\}$

Assert ¬q $\rightsquigarrow \{p, \neg q\}$ $\quad\vdash \{\neg r \vee s, \neg r \vee \neg s\}$

DPLL - Example

Init. $\rightsquigarrow \{\}$ $\quad\vdash \{p \lor q, q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s, p\}$

Assert p $\rightsquigarrow \{p\}$ $\quad\vdash \{p \lor q, q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s, p\}$

Subsume p $\rightsquigarrow \{p\}$ $\quad\vdash \{q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s\}$

Resolve p $\rightsquigarrow \{p\}$ $\quad\vdash \{q \lor \neg r \lor s, \neg q, \neg r \lor \neg s\}$

Assert ¬q $\rightsquigarrow \{p, \neg q\}$ $\quad\vdash \{\neg r \lor s, \neg r \lor \neg s\}$

Split r $\rightsquigarrow \{p, \neg q, r\}$ $\quad\vdash \{s, \neg s\}$

DPLL - Example

Init. $\rightsquigarrow \{\}$ $\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$

Assert p $\rightsquigarrow \{p\}$ $\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$

Subsume p $\rightsquigarrow \{p\}$ $\vdash \{q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s\}$

Resolve p $\rightsquigarrow \{p\}$ $\vdash \{q \vee \neg r \vee s, \neg q, \neg r \vee \neg s\}$

Assert ¬q $\rightsquigarrow \{p, \neg q\}$ $\vdash \{\neg r \vee s, \neg r \vee \neg s\}$

Split r $\rightsquigarrow \{p, \neg q, r\}$ $\vdash \{s, \neg s\}$

DPLL - Example

Init. $\rightsquigarrow \{\}$	$\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$
Assert p $\rightsquigarrow \{p\}$	$\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$
Subsume p $\rightsquigarrow \{p\}$	$\vdash \{q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s\}$
Resolve p $\rightsquigarrow \{p\}$	$\vdash \{q \vee \neg r \vee s, \neg q, \neg r \vee \neg s\}$
Assert ¬q $\rightsquigarrow \{p, \neg q\}$	$\vdash \{\neg r \vee s, \neg r \vee \neg s\}$
Split r $\rightsquigarrow \{p, \neg q, r\}$	$\vdash \{s, \neg s\}$
Split s $\rightsquigarrow \{p, \neg q, r, s\}$	$\vdash \{\Box\}$

DPLL - Example

Init. $\rightsquigarrow \{\}$	$\vdash \{p \lor q, q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s, p\}$
Assert p $\rightsquigarrow \{p\}$	$\vdash \{p \lor q, q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s, p\}$
Subsume p $\rightsquigarrow \{p\}$	$\vdash \{q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s\}$
Resolve p $\rightsquigarrow \{p\}$	$\vdash \{q \lor \neg r \lor s, \neg q, \neg r \lor \neg s\}$
Assert ¬q $\rightsquigarrow \{p, \neg q\}$	$\vdash \{\neg r \lor s, \neg r \lor \neg s\}$
Split r $\rightsquigarrow \{p, \neg q, r\}$	$\vdash \{s, \neg s\}$
Split s $\rightsquigarrow \{p, \neg q, r, s\}$	$\vdash \{\Box\}$
Close	**Contradiction**

DPLL - Example

Init. $\rightsquigarrow \{\}$	$\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$
Assert p $\rightsquigarrow \{p\}$	$\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$
Subsume p $\rightsquigarrow \{p\}$	$\vdash \{q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s\}$
Resolve p $\rightsquigarrow \{p\}$	$\vdash \{q \vee \neg r \vee s, \neg q, \neg r \vee \neg s\}$
Assert ¬q $\rightsquigarrow \{p, \neg q\}$	$\vdash \{\neg r \vee s, \neg r \vee \neg s\}$
Split r $\rightsquigarrow \{p, \neg q, r\}$	$\vdash \{s, \neg s\}$
Split s $\rightsquigarrow \{p, \neg q, r, s\}$	$\vdash \{\square\}$
Close	**Contradiction**

Backtrack $\rightsquigarrow$ **Undo the last Split decision**

DPLL - Example

Init. $\leadsto \{\}$	$\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$
Assert p $\leadsto \{p\}$	$\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$
Subsume p $\leadsto \{p\}$	$\vdash \{q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s\}$
Resolve p $\leadsto \{p\}$	$\vdash \{q \vee \neg r \vee s, \neg q, \neg r \vee \neg s\}$
Assert ¬q $\leadsto \{p, \neg q\}$	$\vdash \{\neg r \vee s, \neg r \vee \neg s\}$
Split r $\leadsto \{p, \neg q, r\}$	$\vdash \{s, \neg s\}$
Split ¬s $\leadsto \{p, \neg q, r, \neg s\}$	$\vdash \{\Box\}$

DPLL - Example

Init. $\rightsquigarrow \{\}$	$\vdash \{p \lor q, q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s, p\}$
Assert p $\rightsquigarrow \{p\}$	$\vdash \{p \lor q, q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s, p\}$
Subsume p $\rightsquigarrow \{p\}$	$\vdash \{q \lor \neg r \lor s, \neg p \lor \neg q, \neg p \lor \neg r \lor \neg s\}$
Resolve p $\rightsquigarrow \{p\}$	$\vdash \{q \lor \neg r \lor s, \neg q, \neg r \lor \neg s\}$
Assert ¬q $\rightsquigarrow \{p, \neg q\}$	$\vdash \{\neg r \lor s, \neg r \lor \neg s\}$
Split r $\rightsquigarrow \{p, \neg q, r\}$	$\vdash \{s, \neg s\}$
Split ¬s $\rightsquigarrow \{p, \neg q, r, \neg s\}$	$\vdash \{\Box\}$
Close	**Contradiction**

DPLL - Example

Init. $\leadsto \{\}$	$\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$
Assert p $\leadsto \{p\}$	$\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$
Subsume p $\leadsto \{p\}$	$\vdash \{q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s\}$
Resolve p $\leadsto \{p\}$	$\vdash \{q \vee \neg r \vee s, \neg q, \neg r \vee \neg s\}$
Assert ¬q $\leadsto \{p, \neg q\}$	$\vdash \{\neg r \vee s, \neg r \vee \neg s\}$
Split ¬r $\leadsto \{p, \neg q, \neg r\}$	$\vdash \{\}$

DPLL - Example

Init. $\leadsto \{\}$ $\quad \vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$

Assert p $\leadsto \{p\}$ $\quad \vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$

Subsume p $\leadsto \{p\}$ $\quad \vdash \{q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s\}$

Resolve p $\leadsto \{p\}$ $\quad \vdash \{q \vee \neg r \vee s, \neg q, \neg r \vee \neg s\}$

Assert ¬q $\leadsto \{p, \neg q\}$ $\quad \vdash \{\neg r \vee s, \neg r \vee \neg s\}$

Split ¬r $\leadsto \{p, \neg q, \neg r\}$ $\quad \vdash \{\}$

Satisfiable

DPLL - Example

Init. $\leadsto \{\}$	$\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$
Assert p $\leadsto \{p\}$	$\vdash \{p \vee q, q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s, p\}$
Subsume p $\leadsto \{p\}$	$\vdash \{q \vee \neg r \vee s, \neg p \vee \neg q, \neg p \vee \neg r \vee \neg s\}$
Resolve p $\leadsto \{p\}$	$\vdash \{q \vee \neg r \vee s, \neg q, \neg r \vee \neg s\}$
Assert ¬q $\leadsto \{p, \neg q\}$	$\vdash \{\neg r \vee s, \neg r \vee \neg s\}$
Split ¬r $\leadsto \{p, \neg q, \neg r\}$	$\vdash \{\}$

Satisfiable

The context gives a model for Φ:

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

Inference Rules – Subsume

Propositional Version

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

First-Order Version by Example

Subsume $$\frac{\Lambda,\ P(x,y) \vdash \Phi,\ P(x,x) \vee Q(x)}{\Lambda,\ P(x,y) \vdash \Phi}$$ **if** $P(x,x)$ **instance of** $P(x,y)$

Inference Rules – Resolve

Propositional Version

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

First-Order Version by Example

$$\textbf{Resolve} \quad \frac{\Lambda,\ P(x,b) \vdash \Phi,\ \neg P(a,y) \vee Q(a)}{\Lambda,\ P(x,b) \vdash \Phi, Q(a)} \quad \textbf{if “see paper”}$$

Inference Rules – Assert

Propositional Version

$$\textbf{Assert}\quad \frac{\Lambda \quad\;\; \vdash \Phi,\; L}{\Lambda, L \vdash \Phi,\; L} \quad \textbf{if} \begin{cases} L \notin \Lambda, \\ \overline{L} \notin \Lambda \end{cases}$$

First-Order Version by Example

$$\textbf{Assert}\quad \frac{\Lambda \qquad\qquad\qquad \vdash \Phi,\; \neg P(x,y)}{\Lambda, \neg P(x,y) \vdash \Phi,\; \neg P(x,y)} \quad \textbf{if “see paper”}$$

Inference Rules – Split

Propositional Version

$$\textbf{Split}\quad \frac{\Lambda \vdash \Phi,\ L \vee C}{\Lambda, L \vdash \Phi,\ L \vee C \qquad \Lambda, \overline{L} \vdash \Phi,\ L \vee C} \quad \textbf{if} \begin{cases} C \neq \Box, \\ L \notin \Lambda, \\ \overline{L} \notin \Lambda \end{cases}$$

First-Order Version by Example

$$\textbf{Split}\quad \frac{\Lambda \vdash \Phi,\ P(x,y) \vee Q(x)}{\Lambda, P(u,y) \vdash \Phi,\ P(x,y) \vee Q(x) \qquad \Lambda, \neg P(u,c) \vdash \Phi,\ P(x,y) \vee Q(x)} \quad \textbf{if}\ \ldots$$

(u schematic variable, y universal variable, c fresh Skolem constant)

Inference Rules – Close

Propositional Version

Close $$\frac{\Lambda \vdash \Phi,\ L_1 \vee \cdots \vee L_n}{\Lambda \vdash \Box}$$ **if** $\begin{cases} \Phi \neq \emptyset \text{ or } n > 0, \\ \overline{L_1}, \ldots, \overline{L_n} \in \Lambda \end{cases}$

First-Order Version by Example

Close $$\frac{\Lambda, \neg P(u,v), \neg Q(x,a) \vdash \Phi,\ P(y,y) \vee Q(z,z)}{\Lambda, \neg P(u,v), \neg Q(x,a) \vdash \Box}$$ **if "see paper"**

(u, v schematic variable, x universal variable)

Context Unifier

- **A context unifier is a simultaneous unifier of a clause from Φ against a multiset of literals from Λ.**

Context Unifier

- **A context unifier is a simultaneous unifier of a clause from Φ against a multiset of literals from Λ.**
- **Example**

  unify clause $P(x, y), Q(x)$

  with context literals $\neg P(u, v), \neg Q(a)$

Context Unifier

- **A context unifier is a simultaneous unifier of a clause from Φ against a multiset of literals from Λ.**
- **Example**

unify clause	$P(x, y), Q(x)$
with context literals	$\neg P(u, v), \neg Q(a)$
context unifier	$\sigma = [x \mapsto a; u \mapsto a; y \mapsto v]$

Context Unifier

- **A context unifier is a simultaneous unifier of a clause from Φ against a multiset of literals from Λ.**
- **Example**

unify clause	$P(x, y), Q(x)$
with context literals	$\neg P(u, v), \neg Q(a)$
context unifier	$\sigma = [x \mapsto a; u \mapsto a; y \mapsto v]$
clause instance	$P(a, v), Q(a)$

Context Unifier

- **A context unifier is a simultaneous unifier of a clause from Φ against a multiset of literals from Λ.**
- **Example**

unify clause $P(x, y), Q(x)$

with context literals $\neg P(u, v), \neg Q(a)$

---

context unifier $\sigma = [x \mapsto a; u \mapsto a; y \mapsto v]$

---

clause instance $\underbrace{P(a, v)}_{remainder}, Q(a)$

Context Unifier

- **A context unifier is a simultaneous unifier of a clause from Φ against a multiset of literals from Λ.**
- **Example**

unify clause	$P(x, y), Q(x)$
with context literals	$\neg P(u, v), \neg Q(a)$
context unifier	$\sigma = [x \mapsto a; u \mapsto a; y \mapsto v]$
clause instance	$\underbrace{P(a, v)}_{remainder}, Q(a)$

- **A Split literal must be from the remainder.**

Context Unifier

- **A context unifier is a simultaneous unifier of a clause from Φ against a multiset of literals from Λ.**
- **Example**

unify clause	$P(x, y), Q(x)$
with context literals	$\neg P(u, v), \neg Q(a)$
context unifier	$\sigma = [x \mapsto a; u \mapsto a; y \mapsto v]$
clause instance	$\underbrace{P(a, v)}_{\textit{remainder}}, Q(a)$

- **A Split literal must be from the remainder.**
- **Close is triggered when the remainder is empty.**

Proof Procedure

```
function start Φ
   // input: a clause set Φ
   // output: either "unsatisfiable"
   //             or a set of literals encoding a model of Φ
   let Λ = ∅     // set of literals
   let L = ¬v // (pseudo) literal
   let Candidates = set of assert literals
                              consisting of the unit clauses in Φ
   try me(Φ,Λ,L,Candidates)
   catch CLOSED -> "unsatisfiable"
```

Proof Procedure

```
function me(Φ, Λ, L, Candidates)
  let Candidates' = add_candidates(Φ, Λ, L, Candidates)
  let Φ' = Φ simplified by Subsume and Resolve
  if no valid candidate in Candidates' then
    Λ'        // Λ' encodes a model of Φ'
  else
    let L' = select_best(Candidates', Λ')
    if L' is an assert literal then
      me(Φ', Λ', L', Candidates' \ {L'})          // assert L
    else
      try
        me(Φ', Λ', L', Candidates' \ {L'})        // left split on L
      catch CLOSED ->
        me(Φ', Λ', ‾L'^sko, Candidates' \ {L'})   // right split on L
```

Proof Procedure

function ***add_candidates***(Φ, Λ, L, $Candidates$)

based on all context unifiers involving L

- adds to $Candidates$ **all assert literals**

- adds to $Candidates$ **one split literal from each remainder**

- raises the exception CLOSED if there is a closing context unifier

function ***select_best***($Candidates$, Λ)

returns the best assert or split literal in $Candidates$

Implementation I

- **Implemented in OCaml.**
- **Iterative Deepening over the term depth of candidate literals.**
- **Default interpretation can alternatively assign true to all atoms.**

Implementation II

- Term indexing with discrimination and substitution trees.
- Term sharing with a term database (set of weak references).
- Bad candidates are stored in a compact format.

Candidate Selection

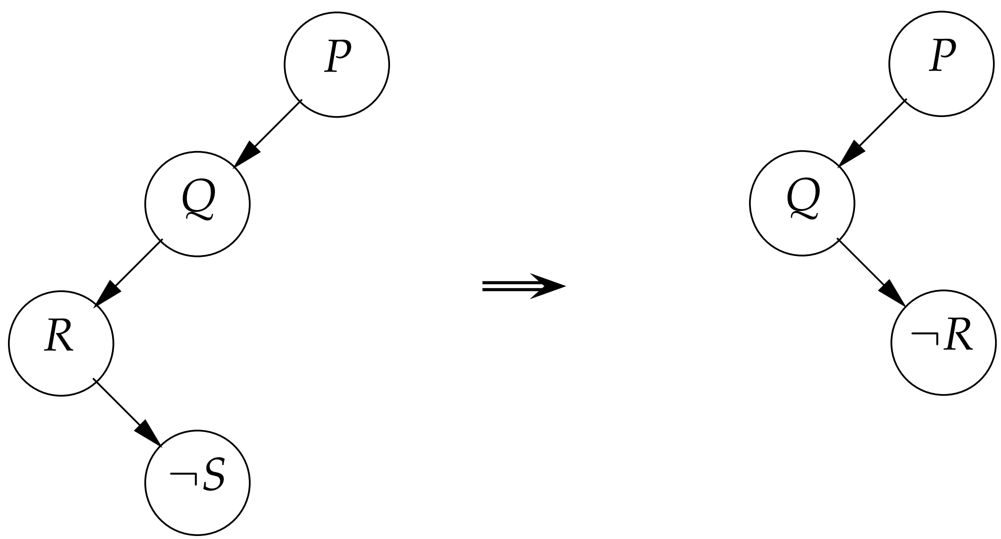
- **Prefer Assert to Split candidates.**
- **Heuristic criteria**
  - **Universality: $p(x)$ and $p(a)$ are preferred to $p(u)$.**
  - **Remainder Size: $p(a)$ is preferred to $p(b) \vee q(c)$.**
  - **Term Weight: $p(x)$ is preferred to $p(f(a))$.**
  - **Generation: Prefer candidates inferred with less derivation steps.**

Partial Context Unifier

- Context unifiers are exhaustively computed per new context literal.
- A context unifier is a simultaneous unifier of a clause from Φ against a multiset of literals from Λ.
- Unifiers between each context and problem literal are cached (partial context unifier).
- A context unifier is computed by merging partial context unifiers.

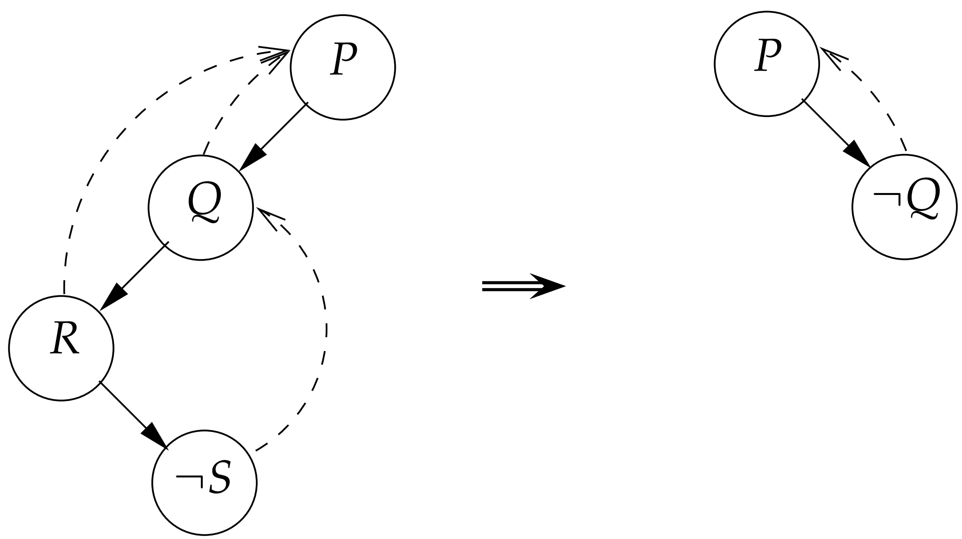
Dependency-Directed Backtracking

- Backtracking replaces left Splits by corresponding right Splits.
- Dependencies between Splits are based on context literals used in corresponding context unifiers.
- Naive Chronological Backtracking



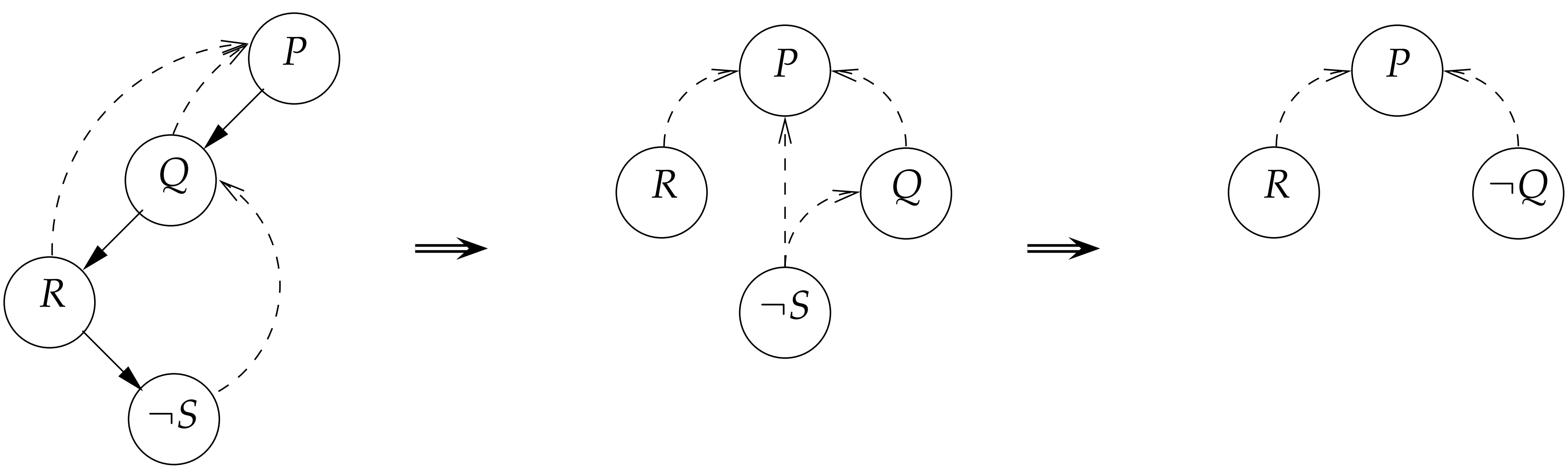
Dependency-Directed Backtracking

- **Backtracking replaces left Splits by corresponding right Splits.**
- **Dependencies between Splits are based on context literals used in corresponding context unifiers.**
- **Backjumping**



Dependency-Directed Backtracking

- **Backtracking replaces left Splits by corresponding right Splits.**
- **Dependencies between Splits are based on context literals used in corresponding context unifiers.**
- **Dynamic Backtracking**



TPTP

- Darwin has been tested with TPTP problems on a P4-2.4Ghz (timeout of 500s, memory limit 500MB).

- Darwin solves
  - 600 of 753 Horn problems without equality
  - 810 of 1172 non-Horn problems without equality

CASC-18

Name	# Probl.	Darwin	DCTP 10.1p/SAT	E-SETHEO csp02/SAT	Gandalf c-2.5/SAT	Vampire 5.0
HNE	35	18	16	28	28	33
HEQ	35	9	20	33	29	31
EPS	35	30	–	27	24	6
EPT	35	32	25	33	34	26
NNE	35	16	22	29	29	33
SNE	35	8	17	19	28	–

HNE – Horn with No Equality

HEQ – Horn with some (but not pure) Equality

EPS – Effectively Propositional non-theorems (satisfiable clause sets)

EPT – Effectively Propositional Theorems (unsatisfiable clause sets)

NNE – Non-Horn with No Equality

SAT with No Equality

CASC-19

Name	# Probl.	Darwin	DCTP 10.2p/SAT	E-SETHEO csp03/SAT	Gandalf c-2.6/SAT	Vampire 6.0
HNE	20	9	15	17	18	18
HEQ	20	0	2	17	10	14
EPS	35	31	34	26	28	15
EPT	35	30	30	31	33	32
NNE	20	9	13	15	13	18
SNE	35	3	17	20	34	–

HNE – Horn with No Equality

HEQ – Horn with some (but not pure) Equality

EPS – Effectively Propositional non-theorems (satisfiable clause sets)

EPT – Effectively Propositional Theorems (unsatisfiable clause sets)

NNE – Non-Horn with No Equality

SAT with No Equality

Evaluation

- **Darwin is a first basic implementation of the ME calculus.**
- **Darwin successfully combines**
  - **liftings of SAT techniques: unit propagation, backjumping**
  - **first-order techniques: unification, subsumption**
  - **theorem prover techniques: term indexing, term database**
- **Darwin already shows very good results for EP problems.**
- **ME leads to a competitive implementation.**

Further Work

- Iterative Deepening over Term Weight, Derivation Tree Depth
- Candidate selection based on conflict sets
- Lemma Learning
- Equality