# Synthesising High-Level Constructs for Set-Based Local Search

Magnus Ågren, Pierre Flener, Justin Pearson

Information Technology, Uppsala University

{agren,pierref,justin}@it.uu.se

SweConsNet'06 – March 15, 2006

# Motivation (1)

We introduced set variables and set constraints in local search.
(See our CPAIOR 2005 paper.)

**Examples:**

- $S \subset T$
- $AllDisjoint(\{S_1, \ldots, S_n\})$
- $MaxIntersect(\{S_1, \ldots, S_n\}, a)$

- Already addressed in constructive search: Gervet, Puget, Müller and Müller.

- Modelling and solving benefits.

# Motivation (2)

- Limited number of implemented set constraints.
- A new (set) constraint in local search requires one (at least):
  - to define penalty and conflict functions for the constraint.
  - to implement incremental maintenance algorithms for penalties and conflicts.
- A time-consuming and error-prone task!

# Idea

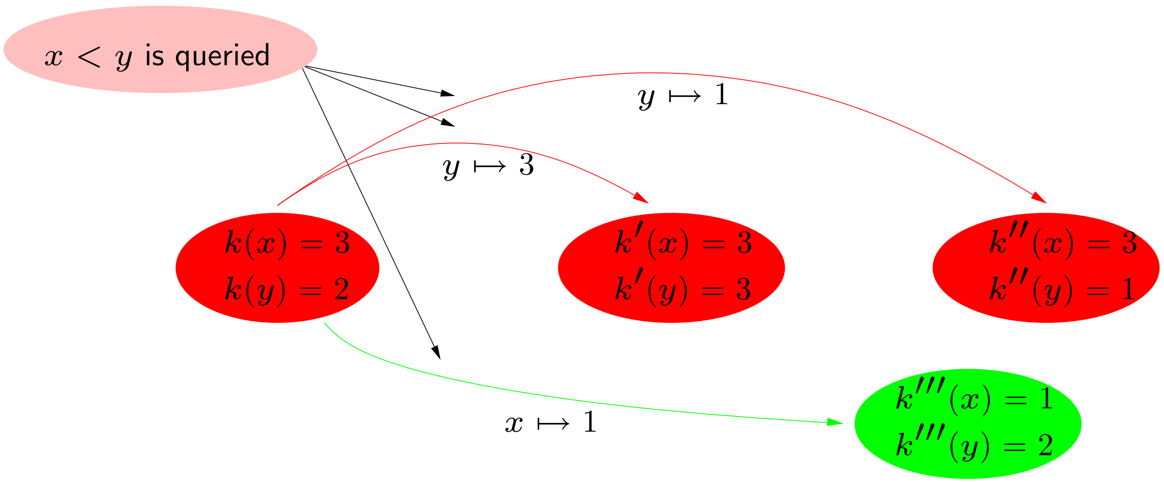
- A modelling language for set constraints.
  - Extend the idea of combinators [Van Hentenryck, Michel & Liu 2004] to quantifiers and set variables.
  - Penalty and conflict functions need only be defined once.
  - Incremental maintenance algorithms need only be implemented once.

- Existential Second-Order Logic ($\exists\mathrm{SOL}$).
  - Small and simple, yet expressive language.
  - Captures at least the complexity class NP.

# Local Search

- Start from a complete assignment (configuration) and iteratively move to promising neighbouring configurations until a (good enough) solution is found.
- Constraints are used to guide the search in the right direction.

**Example:** $\langle\{x \in \{1,2,3,4\}, y \in \{1,2,3\}\}, \{x < y\}\rangle$



# Set Variables

- The domain $D_S$ of a set variable $S$ is a power-set of values, i.e., $D_S = 2^{\mathcal{U}_S}$.
- $\mathcal{U}_S$ is called the universe of $S$.

**Examples:**

$$\mathcal{U}_{S_1} = \mathcal{U}_{S_2} = \{1, 2, 3\}, \mathcal{U}_{S_3} = \{7, 12, 193\}$$

$$k(S_1) = \{2, 3\},\ k(S_2) = \emptyset,\ k(S_3) = \{7\}$$

# Penalty of a Constraint

**Definition 1.** A penalty function of a constraint $c$ is a function $penalty(c) : \mathcal{K} \rightarrow \mathbb{N}$ s.t. $penalty(c)(k) = 0$ if and only if $c$ is satisfied w.r.t. $k$.

**Examples:**

- $penalty(x \leq y)(k) = \max(k(x) - k(y), 0)$
- $penalty(AllDifferent(\mathcal{X}))(k) =$ *"Number of repeated values in $\mathcal{X}$ w.r.t. $k$"*

# Penalty of $S \subset T$

$$penalty(S \subset T)(k) = |k(S) \setminus k(T)| + \begin{cases} 1, & \text{if } k(T) \subseteq k(S) \\ 0, & \text{otherwise} \end{cases}$$

**Examples:**

$k_1(S) = k_1(T) = \{a\}$ gives $penalty(S \subset T)(k_1) = 1$

$k_2(S) = \{a\}, k_2(T) = \emptyset$ gives $penalty(S \subset T)(k_2) = 2$

$k_3(S) = \emptyset, k_3(T) = \{a\}$ gives $penalty(S \subset T)(k_3) = 0$

# Existential Second-Order Logic (with Counting)

## BNF grammar of $\exists\mathrm{SOL}^+$

$$
\begin{array}{rcl}
\langle Constraint \rangle & ::= & (\underline{\exists}\ \langle S \rangle)^+\ \langle Formula \rangle \\
\langle Formula \rangle & ::= & \underline{(}\langle Formula \rangle\underline{)} \\
& \mid & (\underline{\forall} \mid \underline{\exists})\langle x \rangle\ \langle Formula \rangle \\
& \mid & \langle Formula \rangle\ (\underline{\wedge} \mid \underline{\vee})\ \langle Formula \rangle \\
& \mid & \langle Literal \rangle \\
& \mid & \langle Formula \rangle\ (\underline{\rightarrow} \mid \underline{\leftrightarrow} \mid \underline{\leftarrow})\ \langle Formula \rangle \\
& \mid & \underline{\neg}\langle Formula \rangle \\
\langle Literal \rangle & ::= & \langle x \rangle\ (\underline{\in} \mid \underline{\notin})\ \langle S \rangle \\
& \mid & \langle x \rangle\ (\underline{<} \mid \underline{\leq} \mid \underline{=} \mid \underline{\neq} \mid \underline{\geq} \mid \underline{>})\ \langle y \rangle \\
& \mid & \underline{|}\langle S \rangle\underline{|}\ (\underline{<} \mid \underline{\leq} \mid \underline{=} \mid \underline{\neq} \mid \underline{\geq} \mid \underline{>})\ \langle a \rangle
\end{array}
$$

## $\exists\mathrm{SOL}^+$

- A sequence of $\exists$-quantified set variables constrained by a logical formula.
- All set variables share the same universe $\mathcal{U}$.
- Negation as well as implications removed.

# Existential Second-Order Logic (with Counting)

## BNF grammar of $\exists\mathrm{SOL}^{+}$

$$
\begin{aligned}
\langle \mathit{Constraint} \rangle &::= (\underline{\exists}\ \langle S \rangle)^{+}\ \langle \mathit{Formula} \rangle \\
\langle \mathit{Formula} \rangle &::= \underline{(}\langle \mathit{Formula} \rangle\underline{)} \\
&\quad \mid (\underline{\forall} \mid \underline{\exists})\langle x \rangle\ \langle \mathit{Formula} \rangle \\
&\quad \mid \langle \mathit{Formula} \rangle\ (\underline{\wedge} \mid \underline{\vee})\ \langle \mathit{Formula} \rangle \\
&\quad \mid \langle \mathit{Literal} \rangle \\
&\quad \mid \langle \mathit{Formula} \rangle\ (\underline{\rightarrow} \mid \underline{\leftrightarrow} \mid \underline{\leftarrow})\ \langle \mathit{Formula} \rangle \\
&\quad \mid \underline{\neg}\langle \mathit{Formula} \rangle \\
\langle \mathit{Literal} \rangle &::= \langle x \rangle\ (\underline{\in} \mid \underline{\notin})\ \langle S \rangle \\
&\quad \mid \langle x \rangle\ (\underline{<} \mid \underline{\leq} \mid \underline{=} \mid \underline{\neq} \mid \underline{\geq} \mid \underline{>})\ \langle y \rangle \\
&\quad \mid \underline{|}\langle S \rangle\underline{|}\ (\underline{<} \mid \underline{\leq} \mid \underline{=} \mid \underline{\neq} \mid \underline{\geq} \mid \underline{>})\ \langle a \rangle
\end{aligned}
$$

## $S \subset T$ in $\exists\mathrm{SOL}^{+}$

$$S \subset T$$

$$\Leftrightarrow$$

$$\exists S \exists T((\forall x(x \notin S \vee x \in T)) \wedge (\exists x(x \in T \wedge x \notin S)))$$

Quantification over the whole $\mathcal{U}$.

# Inductive Definition: Penalty of an $\exists\mathrm{SOL}^+$ Formula

$$
\begin{aligned}
\mathit{penalty}(\exists S_1 \cdots \exists S_n \phi)(k) &= \mathit{penalty}(\phi)(k) \\
\mathit{penalty}(\forall x \phi)(k) &= \sum_{u \in \mathcal{U}} \mathit{penalty}(\phi)(k \cup \{x \mapsto u\}) \\
\mathit{penalty}(\exists x \phi)(k) &= \min\{\mathit{penalty}(\phi)(k \cup \{x \mapsto u\} \mid u \in \mathcal{U}\}) \\
\mathit{penalty}(\phi \wedge \psi)(k) &= \mathit{penalty}(\phi)(k) + \mathit{penalty}(\psi)(k) \\
\mathit{penalty}(\phi \vee \psi)(k) &= \min\{\mathit{penalty}(\phi)(k), \mathit{penalty}(\psi)(k)\} \\
\mathit{penalty}(|S| \leq c)(k) &= \begin{cases} 0, & \text{if } |k(S)| \leq c \\ |k(S)| - c, & \text{otherwise} \end{cases} \\
\mathit{penalty}(x \in S)(k) &= \begin{cases} 0, & \text{if } k(x) \in k(S) \\ 1, & \text{otherwise} \end{cases} \\
\mathit{penalty}(x \leq y)(k) &= \begin{cases} 0, & \text{if } k(x) \leq k(y) \\ 1, & \text{otherwise} \end{cases}
\end{aligned}
$$

# Penalty of the $S \subset T$ Formula

$$\mathcal{F} = \exists S \exists T(\underbrace{(\forall x(x \notin S \vee x \in T))}_{\mathcal{F}_1} \wedge \underbrace{(\exists x(x \in T \wedge x \notin S))}_{\mathcal{F}_2})$$

$$\mathcal{U} = \{a, b\},\ k(S) = \{a\}, k(T) = \emptyset$$

# Penalty of the $S \subset T$ Formula

$$\mathcal{F} = \exists S \exists T(\underbrace{(\forall x(x \notin S \lor x \in T))}_{\mathcal{F}_1} \land \underbrace{(\exists x(x \in T \land x \notin S))}_{\mathcal{F}_2})$$

$\mathcal{U} = \{a, b\}$, $k(S) = \{a\}, k(T) = \emptyset$

1. $p(\mathcal{F})(k) = p(\mathcal{F}_1)(k) + p(\mathcal{F}_2)(k)$

# Penalty of the $S \subset T$ Formula

$$\mathcal{F} = \exists S \exists T (\underbrace{(\forall x (x \notin S \lor x \in T))}_{\mathcal{F}_1} \land \underbrace{(\exists x (x \in T \land x \notin S))}_{\mathcal{F}_2})$$

$\mathcal{U} = \{a, b\}$, $k(S) = \{a\}, k(T) = \emptyset$

1. $p(\mathcal{F})(k) = p(\mathcal{F}_1)(k) + p(\mathcal{F}_2)(k)$
2. $p(\mathcal{F}_1)(k) = p(a \notin S \lor a \in T)(k) +$
   $p(b \notin S \lor b \in T)(k)$

# Penalty of the $S \subset T$ Formula

$$\mathcal{F} = \exists S \exists T (\underbrace{(\forall x (x \notin S \lor x \in T))}_{\mathcal{F}_1} \land \underbrace{(\exists x (x \in T \land x \notin S))}_{\mathcal{F}_2})$$

$\mathcal{U} = \{a, b\}$, $k(S) = \{a\}, k(T) = \emptyset$

1. $p(\mathcal{F})(k) = p(\mathcal{F}_1)(k) + p(\mathcal{F}_2)(k)$
2. $p(\mathcal{F}_1)(k) = p(a \notin S \lor a \in T)(k) +$
   $p(b \notin S \lor b \in T)(k)$
3. $p(a \notin S \lor a \in T)(k) =$
   $\min(p(a \notin S)(k), p(a \in T)(k)) =$
   $\min(1, 1) = 1$

# Penalty of the $S \subset T$ Formula

$$\mathcal{F} = \exists S \exists T (\underbrace{(\forall x (x \notin S \lor x \in T))}_{\mathcal{F}_1} \land \underbrace{(\exists x (x \in T \land x \notin S))}_{\mathcal{F}_2})$$

$$\mathcal{U} = \{a, b\},\ k(S) = \{a\}, k(T) = \emptyset$$

1. $p(\mathcal{F})(k) = p(\mathcal{F}_1)(k) + p(\mathcal{F}_2)(k)$
2. $p(\mathcal{F}_1)(k) = p(a \notin S \lor a \in T)(k) +$
   $p(b \notin S \lor b \in T)(k)$
3. $p(a \notin S \lor a \in T)(k) =$
   $\min(p(a \notin S)(k), p(a \in T)(k)) =$
   $\min(1, 1) = 1$
4. $p(b \notin S \lor b \in T)(k) =$
   $\min(p(b \notin S)(k), p(b \in T)(k)) =$
   $\min(0, 1) = 0$

# Penalty of the $S \subset T$ Formula

$$\mathcal{F} = \exists S \exists T(\underbrace{(\forall x(x \notin S \vee x \in T))}_{\mathcal{F}_1} \wedge \underbrace{(\exists x(x \in T \wedge x \notin S))}_{\mathcal{F}_2})$$

$$\mathcal{U} = \{a, b\},\ k(S) = \{a\}, k(T) = \emptyset$$

1. $p(\mathcal{F})(k) = p(\mathcal{F}_1)(k) + p(\mathcal{F}_2)(k)$
2. $p(\mathcal{F}_1)(k) = p(a \notin S \vee a \in T)(k) + p(b \notin S \vee b \in T)(k)$
3. $p(a \notin S \vee a \in T)(k) = \min(p(a \notin S)(k), p(a \in T)(k)) = \min(1, 1) = 1$
4. $p(b \notin S \vee b \in T)(k) = \min(p(b \notin S)(k), p(b \in T)(k)) = \min(0, 1) = 0$
5. $p(\mathcal{F}_2)(k) = \min(p(a \in T \wedge a \notin S)(k), p(b \in T \wedge b \notin S)(k))$

# Penalty of the $S \subset T$ Formula

$$\mathcal{F} = \exists S \exists T(\underbrace{(\forall x(x \notin S \lor x \in T))}_{\mathcal{F}_1} \land \underbrace{(\exists x(x \in T \land x \notin S))}_{\mathcal{F}_2})$$

$\mathcal{U} = \{a, b\}$, $k(S) = \{a\}, k(T) = \emptyset$

1. $p(\mathcal{F})(k) = p(\mathcal{F}_1)(k) + p(\mathcal{F}_2)(k)$
2. $p(\mathcal{F}_1)(k) = p(a \notin S \lor a \in T)(k) + p(b \notin S \lor b \in T)(k)$
3. $p(a \notin S \lor a \in T)(k) = \min(p(a \notin S)(k), p(a \in T)(k)) = \min(1, 1) = 1$
4. $p(b \notin S \lor b \in T)(k) = \min(p(b \notin S)(k), p(b \in T)(k)) = \min(0, 1) = 0$
5. $p(\mathcal{F}_2)(k) = \min(p(a \in T \land a \notin S)(k), p(b \in T \land b \notin S)(k))$
6. $p(a \in T \land a \notin S)(k) = p(a \in T)(k) + p(a \notin S)(k) = 1 + 1 = 2$

# Penalty of the $S \subset T$ Formula

$$\mathcal{F} = \exists S \exists T(\underbrace{(\forall x(x \notin S \lor x \in T))}_{\mathcal{F}_1} \land \underbrace{(\exists x(x \in T \land x \notin S))}_{\mathcal{F}_2})$$

$\mathcal{U} = \{a, b\}$, $k(S) = \{a\}, k(T) = \emptyset$

1. $p(\mathcal{F})(k) = p(\mathcal{F}_1)(k) + p(\mathcal{F}_2)(k)$
2. $p(\mathcal{F}_1)(k) = p(a \notin S \lor a \in T)(k) + p(b \notin S \lor b \in T)(k)$
3. $p(a \notin S \lor a \in T)(k) = \min(p(a \notin S)(k), p(a \in T)(k)) = \min(1, 1) = 1$
4. $p(b \notin S \lor b \in T)(k) = \min(p(b \notin S)(k), p(b \in T)(k)) = \min(0, 1) = 0$
5. $p(\mathcal{F}_2)(k) = \min(p(a \in T \land a \notin S)(k), p(b \in T \land b \notin S)(k))$
6. $p(a \in T \land a \notin S)(k) = p(a \in T)(k) + p(a \notin S)(k) = 1 + 1 = 2$
7. $p(b \in T \land b \notin S)(k) = p(b \in T)(k) + p(b \notin S)(k) = 1 + 0 = 1$

# Penalty of the $S \subset T$ Formula

$$\mathcal{F} = \exists S \exists T (\underbrace{(\forall x(x \notin S \lor x \in T))}_{\mathcal{F}_1} \land \underbrace{(\exists x(x \in T \land x \notin S))}_{\mathcal{F}_2})$$

$\mathcal{U} = \{a, b\}$, $k(S) = \{a\}, k(T) = \emptyset$

1. $p(\mathcal{F})(k) = p(\mathcal{F}_1)(k) + p(\mathcal{F}_2)(k)$
2. $p(\mathcal{F}_1)(k) = p(a \notin S \lor a \in T)(k) + p(b \notin S \lor b \in T)(k)$
3. $p(a \notin S \lor a \in T)(k) = \min(p(a \notin S)(k), p(a \in T)(k)) = \min(1, 1) = 1$
4. $p(b \notin S \lor b \in T)(k) = \min(p(b \notin S)(k), p(b \in T)(k)) = \min(0, 1) = 0$
5. $p(\mathcal{F}_2)(k) = \min(p(a \in T \land a \notin S)(k), 1)$
6. $p(a \in T \land a \notin S)(k) = p(a \in T)(k) + p(a \notin S)(k) = 1 + 1 = 2$
7. $p(b \in T \land b \notin S)(k) = 1$

# Penalty of the $S \subset T$ Formula

$$\mathcal{F} = \exists S \exists T(\underbrace{(\forall x(x \notin S \lor x \in T))}_{\mathcal{F}_1} \land \underbrace{(\exists x(x \in T \land x \notin S))}_{\mathcal{F}_2})$$

$\mathcal{U} = \{a, b\}$, $k(S) = \{a\}, k(T) = \emptyset$

1. $p(\mathcal{F})(k) = p(\mathcal{F}_1)(k) + p(\mathcal{F}_2)(k)$
2. $p(\mathcal{F}_1)(k) = p(a \notin S \lor a \in T)(k) +$ $p(b \notin S \lor b \in T)(k)$
3. $p(a \notin S \lor a \in T)(k) =$ $\min(p(a \notin S)(k), p(a \in T)(k)) =$ $\min(1, 1) = 1$
4. $p(b \notin S \lor b \in T)(k) =$ $\min(p(b \notin S)(k), p(b \in T)(k)) =$ $\min(0, 1) = 0$
5. $p(\mathcal{F}_2)(k) = \min(2, 1)$
6. $p(a \in T \land a \notin S)(k) = 2$
7. $p(b \in T \land b \notin S)(k) = 1$

# Penalty of the $S \subset T$ Formula

$$\mathcal{F} = \exists S \exists T(\underbrace{(\forall x(x \notin S \vee x \in T))}_{\mathcal{F}_1} \wedge \underbrace{(\exists x(x \in T \wedge x \notin S))}_{\mathcal{F}_2})$$

$$\mathcal{U} = \{a, b\},\ k(S) = \{a\}, k(T) = \emptyset$$

1. $p(\mathcal{F})(k) = p(\mathcal{F}_1)(k) + 1$
2. $p(\mathcal{F}_1)(k) = p(a \notin S \vee a \in T)(k) +$ $p(b \notin S \vee b \in T)(k)$
3. $p(a \notin S \vee a \in T)(k) =$ $\min(p(a \notin S)(k), p(a \in T)(k)) =$ $\min(1, 1) = 1$
4. $p(b \notin S \vee b \in T)(k) =$ $\min(p(b \notin S)(k), p(b \in T)(k)) =$ $\min(0, 1) = 0$
5. $p(\mathcal{F}_2)(k) = \min(2, 1) = 1$
6. $p(a \in T \wedge a \notin S)(k) = 2$
7. $p(b \in T \wedge b \notin S)(k) = 1$

# Penalty of the $S \subset T$ Formula

$$\mathcal{F} = \exists S \exists T(\underbrace{(\forall x(x \notin S \lor x \in T))}_{\mathcal{F}_1} \land \underbrace{(\exists x(x \in T \land x \notin S))}_{\mathcal{F}_2})$$

$\mathcal{U} = \{a, b\}$, $k(S) = \{a\}, k(T) = \emptyset$

1. $p(\mathcal{F})(k) = p(\mathcal{F}_1)(k) + 1$
2. $p(\mathcal{F}_1)(k) = p(a \notin S \lor a \in T)(k) +$ $0$
3. $p(a \notin S \lor a \in T)(k) =$ $\min(p(a \notin S)(k), p(a \in T)(k)) =$ $\min(1, 1) = 1$
4. $p(b \notin S \lor b \in T)(k) = 0$
5. $p(\mathcal{F}_2)(k) = \min(2, 1) = 1$
6. $p(a \in T \land a \notin S)(k) = 2$
7. $p(b \in T \land b \notin S)(k) = 1$

# Penalty of the $S \subset T$ Formula

$$\mathcal{F} = \exists S \exists T(\underbrace{(\forall x(x \notin S \vee x \in T))}_{\mathcal{F}_1} \wedge \underbrace{(\exists x(x \in T \wedge x \notin S))}_{\mathcal{F}_2})$$

$\mathcal{U} = \{a, b\}$, $k(S) = \{a\}, k(T) = \emptyset$

1. $p(\mathcal{F})(k) = p(\mathcal{F}_1)(k) + 1$
2. $p(\mathcal{F}_1)(k) = 1 + 0$
3. $p(a \notin S \vee a \in T)(k) = 1$
4. $p(b \notin S \vee b \in T)(k) = 0$
5. $p(\mathcal{F}_2)(k) = \min(2, 1) = 1$
6. $p(a \in T \wedge a \notin S)(k) = 2$
7. $p(b \in T \wedge b \notin S)(k) = 1$

# Penalty of the $S \subset T$ Formula

$$\mathcal{F} = \exists S \exists T(\underbrace{(\forall x(x \notin S \vee x \in T))}_{\mathcal{F}_1} \wedge \underbrace{(\exists x(x \in T \wedge x \notin S))}_{\mathcal{F}_2})$$

$\mathcal{U} = \{a, b\}$, $k(S) = \{a\}, k(T) = \emptyset$

1. $p(\mathcal{F})(k) = 1 + 1$
2. $p(\mathcal{F}_1)(k) = 1 + 0 = 1$
3. $p(a \notin S \vee a \in T)(k) = 1$
4. $p(b \notin S \vee b \in T)(k) = 0$
5. $p(\mathcal{F}_2)(k) = \min(2, 1) = 1$
6. $p(a \in T \wedge a \notin S)(k) = 2$
7. $p(b \in T \wedge b \notin S)(k) = 1$

# Penalty of the $S \subset T$ Formula

$$\mathcal{F} = \exists S \exists T(\underbrace{(\forall x(x \notin S \vee x \in T))}_{\mathcal{F}_1} \wedge \underbrace{(\exists x(x \in T \wedge x \notin S))}_{\mathcal{F}_2})$$

$\mathcal{U} = \{a, b\}$, $k(S) = \{a\}, k(T) = \emptyset$

1. $p(\mathcal{F})(k) = 1 + 1 = 2$
2. $p(\mathcal{F}_1)(k) = 1 + 0 = 1$
3. $p(a \notin S \vee a \in T)(k) = 1$
4. $p(b \notin S \vee b \in T)(k) = 0$
5. $p(\mathcal{F}_2)(k) = \min(2, 1) = 1$
6. $p(a \in T \wedge a \notin S)(k) = 2$
7. $p(b \in T \wedge b \notin S)(k) = 1$

Indeed, exactly two values must be changed in $k(S)$ and/or $k(T)$ to satisfy $k(S) \subset k(T)$.

# Efficiency Issues

- The number of different configurations to explore in a real-life problem may be as large as 500,000,000, if not larger.

- Recalculating from scratch the value of $penalty(c)(k')$ for a constraint $c$ for each neighbouring configuration $k'$ of $k$ is impractical.

- The penalty functions must be defined incrementally.

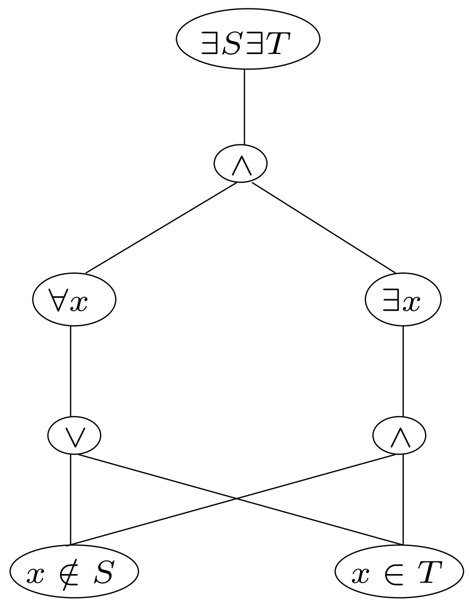
- Two parts of each function $penalty(c)$:
  - $penalty_{init}(c)(k)$
  - $penalty_{delta}(c)(k')$, where $k' = k + \delta$ and $penalty(c)(k)$ is known. (Hence $\delta$ is the difference between $k$ and $k'$.)

# Incremental Penalty Maintenance Using Penalty Trees

## Idea

- Build a syntax tree of an $\exists\mathrm{SOL}^+$ formula.
- Populate the syntax tree with information to obtain a penalty tree.

## Syntax Tree of $S \subset T$

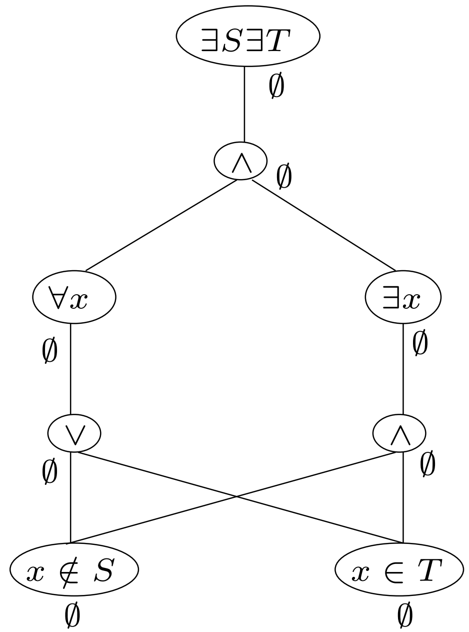


# Incremental Penalty Maintenance Using Penalty Trees

## Idea

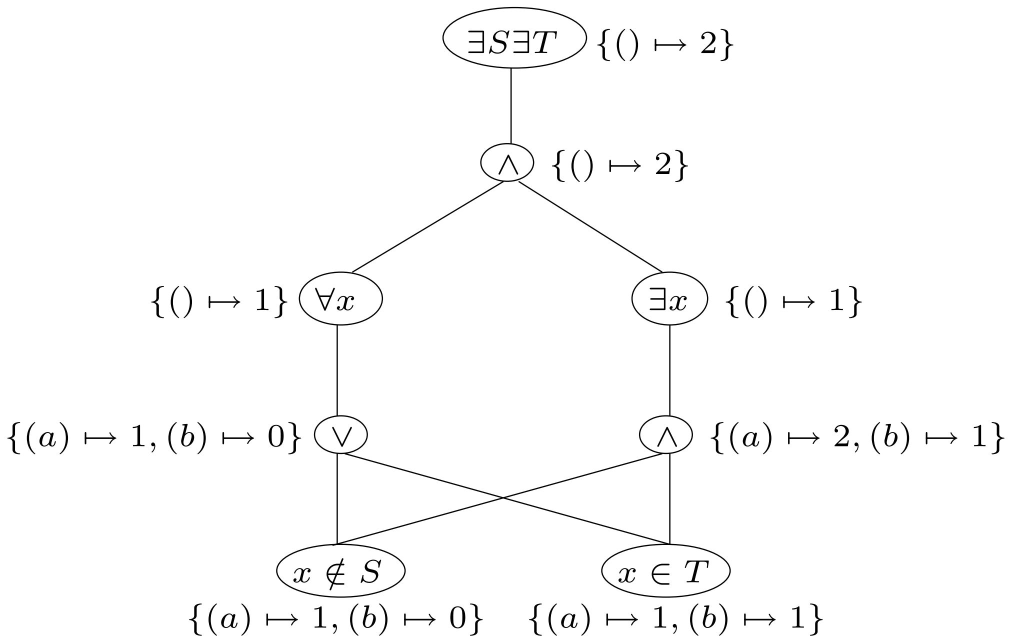
- Build a syntax tree of an $\exists\mathrm{SOL}^+$ formula.
- Populate the syntax tree with information to obtain a penalty tree.

## Penalty Tree of $S \subset T$



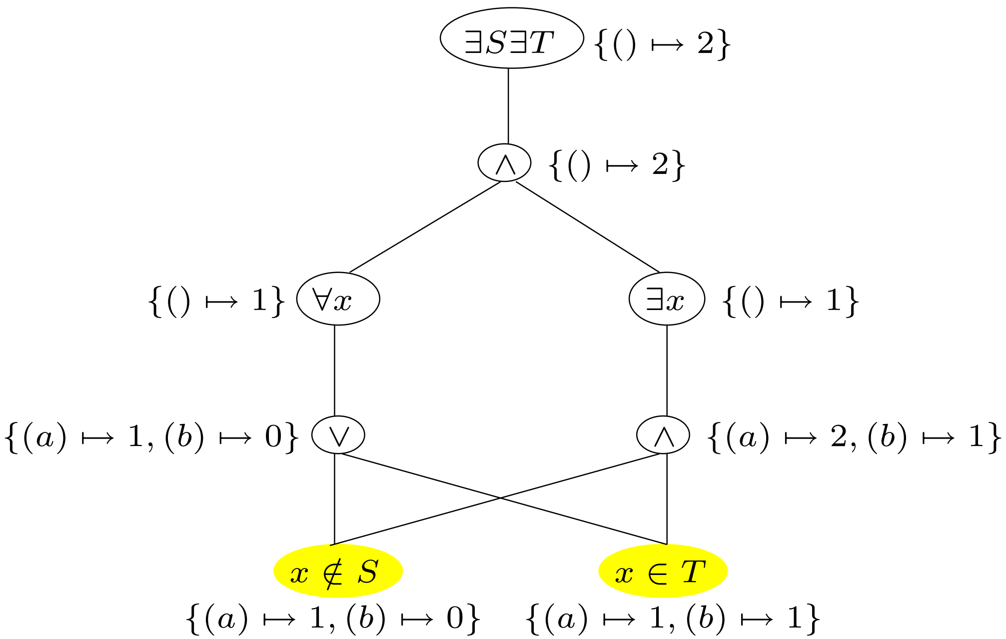
# Initialising the Penalty Tree of $S \subset T$

$$\mathcal{U} = \{a, b\},\ k(S) = \{a\},\ k(T) = \emptyset$$



# Incrementally Updating the Penalty Tree of $S \subset T$

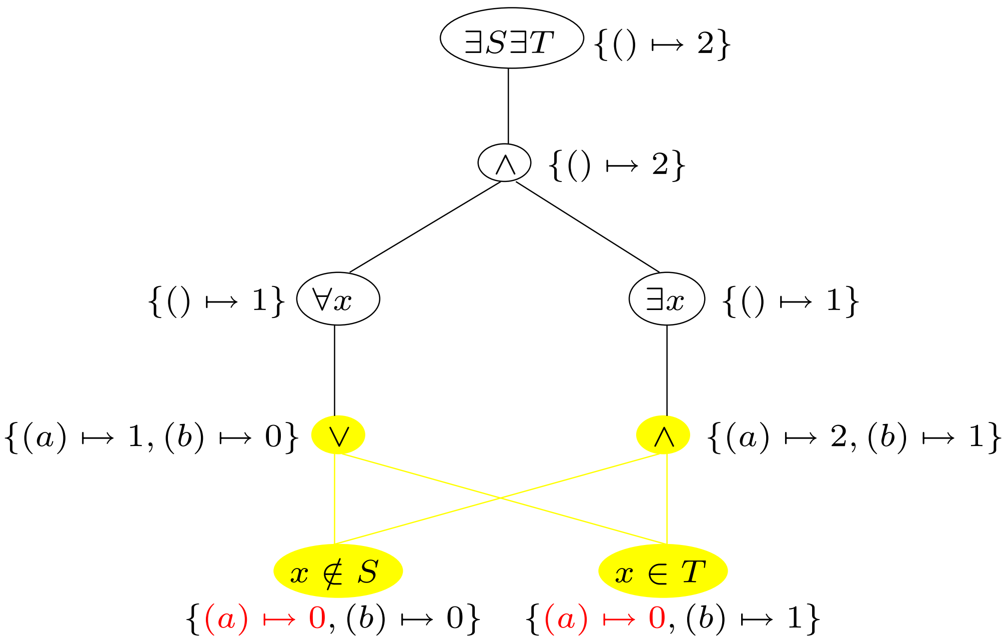
**Change:** Move $a$ from $S$ to $T$.
**New state:** $\mathcal{U} = \{a, b\}$, $k'(S) = \emptyset, k'(T) = \{a\}$




- Only affected paths need updating.
- Start from affected leaves and update paths to the root node.

# Incrementally Updating the Penalty Tree of $S \subset T$

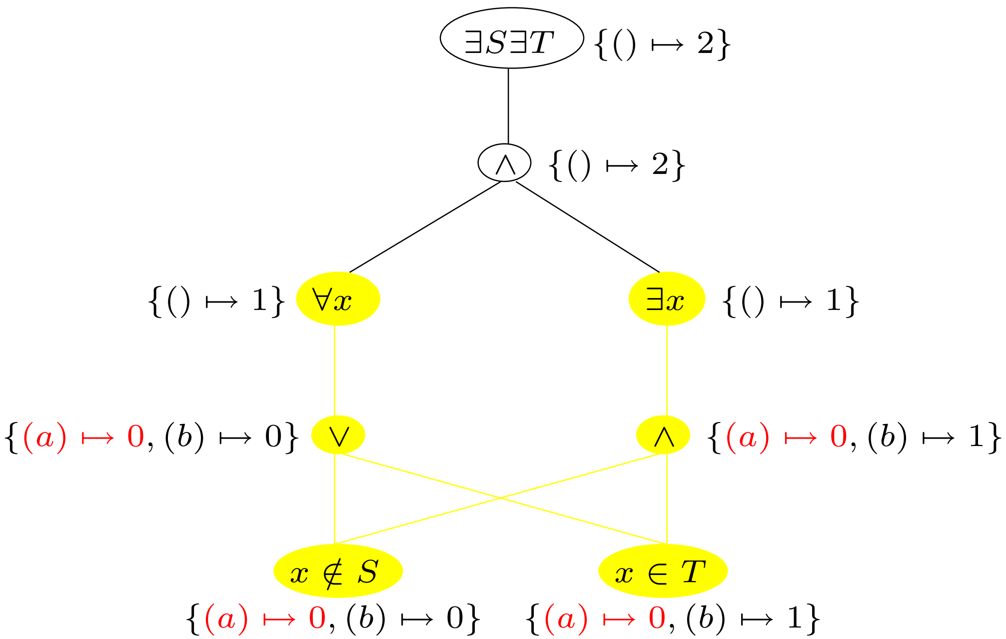
**Change:** Move $a$ from $S$ to $T$.
**New state:** $\mathcal{U} = \{a, b\}$, $k'(S) = \emptyset, k'(T) = \{a\}$




- Only affected paths need updating.
- Start from affected leaves and update paths to the root node.

# Incrementally Updating the Penalty Tree of $S \subset T$

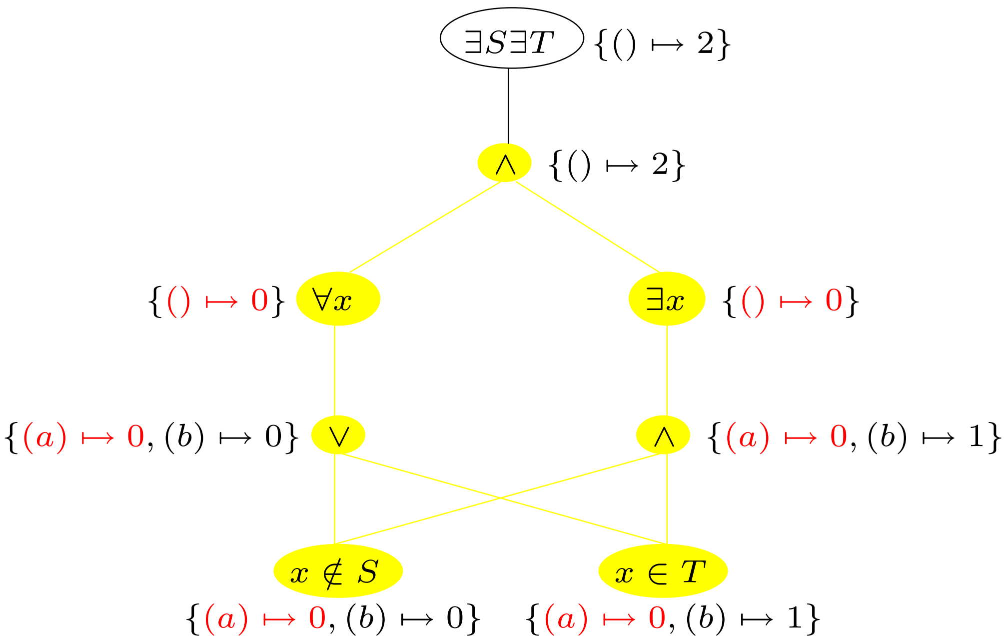
**Change:** Move $a$ from $S$ to $T$.
**New state:** $\mathcal{U} = \{a, b\}$, $k'(S) = \emptyset, k'(T) = \{a\}$




- Only affected paths need updating.
- Start from affected leaves and update paths to the root node.

# Incrementally Updating the Penalty Tree of $S \subset T$

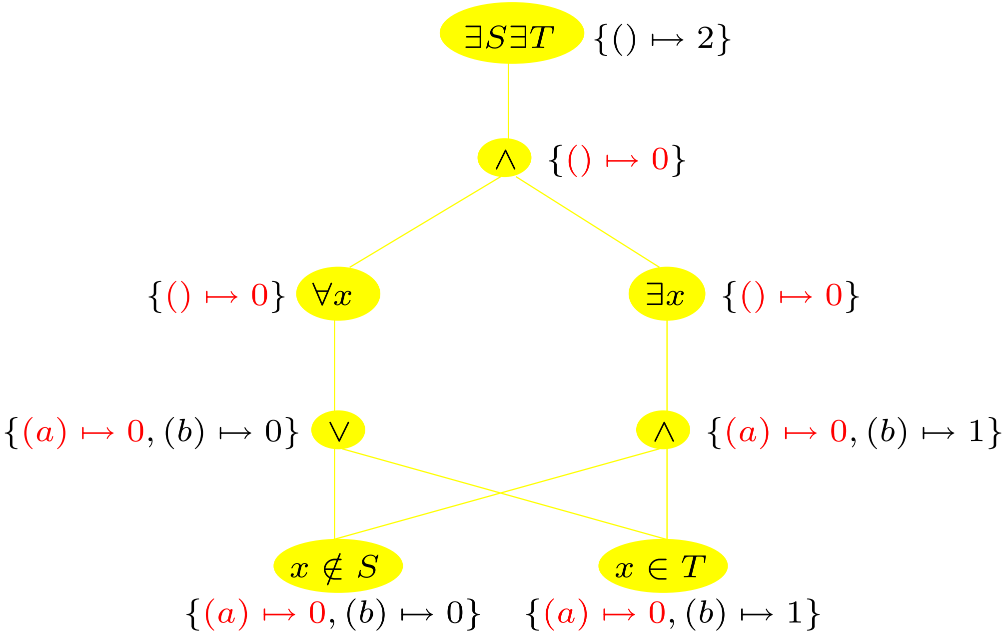
**Change:** Move $a$ from $S$ to $T$.
**New state:** $\mathcal{U} = \{a, b\}$, $k'(S) = \emptyset, k'(T) = \{a\}$




- Only affected paths need updating.
- Start from affected leaves and update paths to the root node.

# Incrementally Updating the Penalty Tree of $S \subset T$

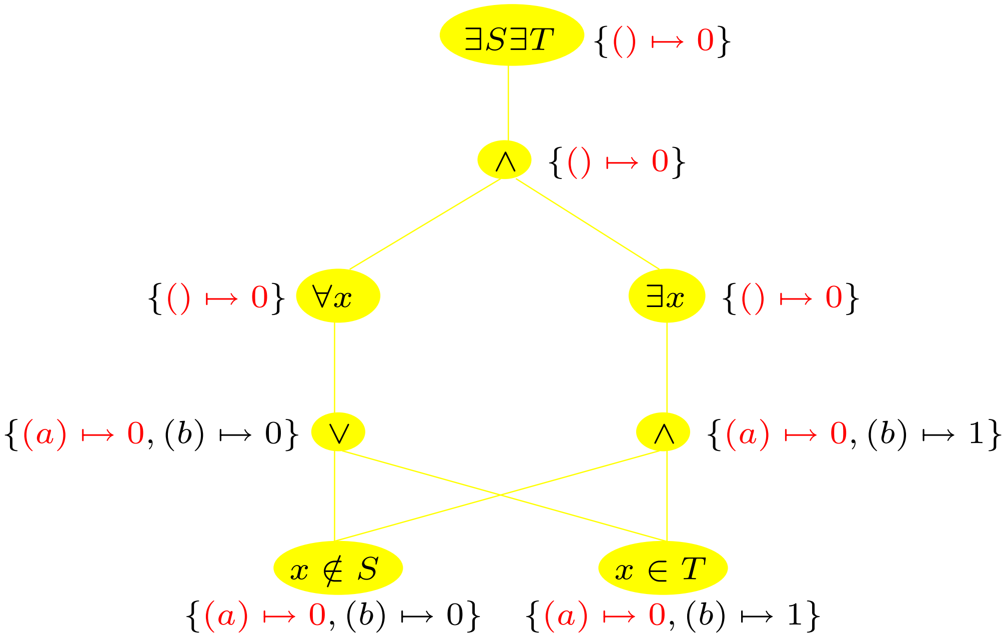
**Change:** Move $a$ from $S$ to $T$.
**New state:** $\mathcal{U} = \{a, b\}$, $k'(S) = \emptyset, k'(T) = \{a\}$




- Only affected paths need updating.
- Start from affected leaves and update paths to the root node.

# Incrementally Updating the Penalty Tree of $S \subset T$

**Change:** Move $a$ from $S$ to $T$.
**New state:** $\mathcal{U} = \{a, b\}$, $k'(S) = \emptyset, k'(T) = \{a\}$




- Only affected paths need updating.
- Start from affected leaves and update paths to the root node.

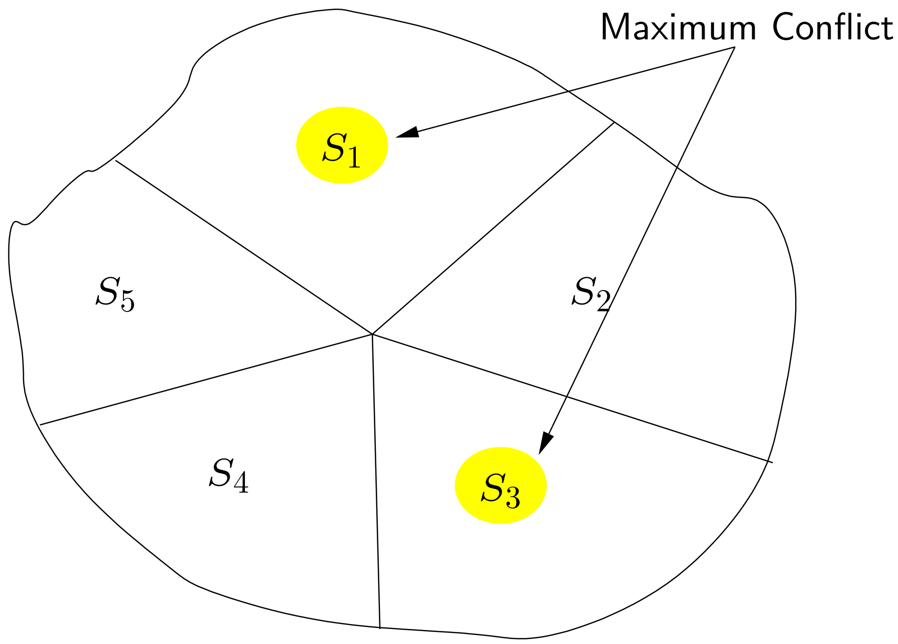
# Conflicting Variables

- A possible neighbourhood (1): *"Move each value in any set to any other set"*
- Impractical in reality!
- Focus on conflicting variables.
- A possible neighbourhood (2): *"Move each value in $S$ to any other set"* where $S$ has the maximum conflict.

# Conflicting Variables

- A possible neighbourhood (1): *"Move each value in any set to any other set"*
- Impractical in reality!
- Focus on conflicting variables.
- A possible neighbourhood (2): *"Move each value in $S$ to any other set"* where $S$ has the maximum conflict.

Splitting the Search Space



# Conflict of a Variable

**Definition 2.** Let $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ be a CSP. A conflict function of $c \in \mathcal{C}$ is a function $\mathit{conflict}(c) : \mathcal{X} \times \mathcal{K} \to \mathbb{N}$ s.t. if $\mathit{conflict}(c)(x,k) = 0$ then $\forall \ell \in \mathcal{N}_x(k) : \mathit{penalty}(c)(k) \leq \mathit{penalty}(c)(\ell)$.

$\mathcal{N}_x(k)$ is the set of configurations reachable from $k$ by only changing $k(x)$.

**Examples:**

$$\mathit{conflict}(x \leq y)(z,k) = \begin{cases} \max(k(x) - k(y), 0), & \text{if } z = x \text{ or } z = y \\ 0, & \text{otherwise} \end{cases}$$

$$\mathit{conflict}(\mathit{AllDifferent}(\mathcal{X}))(x,k) = \begin{cases} 1, & \text{if } x \in \mathcal{X} \ \ \& \ \ \exists\, y \neq x \in \mathcal{X} : k(x) = k(y) \\ 0, & \text{otherwise} \end{cases}$$

# Conflict with respect to $S \subset T$

$$conflict(S \subset T)(Q, k) = |k(S) \setminus k(T)| + \begin{cases} 1, & \text{if } Q = T \text{ and } k(T) \subseteq k(S) \\ 1, & \text{if } Q = S \text{ and } k(S) \neq \emptyset \text{ and } k(T) \subseteq k(S) \\ 0, & \text{otherwise} \end{cases}$$

**Examples:**

**Recall:** $k_2(S) = \{a\}, k_2(T) = \emptyset, penalty(S \subset T)(k_2) = 2$
**Then** $conflict(S \subset T)(S, k_2) = 1$ **and** $conflict(S \subset T)(T, k_2) = 2$

**Recall:** $k_3(S) = \emptyset, k_3(T) = \{a\}, penalty(S \subset T)(k_3) = 0$
**Then** $conflict(S \subset T)(S, k) = 0$ **and** $conflict(S \subset T)(T, k) = 0$

# Inductive Definition: Conflict w.r.t. an $\exists\mathrm{SOL}^+$ Formula

$$conflict(\exists S_1 \cdots \exists S_n \phi)(S, k) = conflict(\phi)(S, k)$$

$$conflict(\forall x \phi)(S, k) = \sum_{u \in \mathcal{U}} conflict(\phi)(S, k \cup \{x \mapsto u\})$$

$$conflict(\exists x \phi)(S, k) = \max\{0\} \cup \{penalty(\exists x \phi)(k) - (penalty(\phi)(k \cup \{x \mapsto u\}) - conflict(\phi)(S, k \cup \{x \mapsto u\})) \mid u \in \mathcal{U}\}$$

$$conflict(\phi \wedge \psi)(S, k) = \sum\{conflict(\gamma)(S, k) \mid \gamma \in \{\phi, \psi\} \wedge S \in vars(\gamma)\}$$

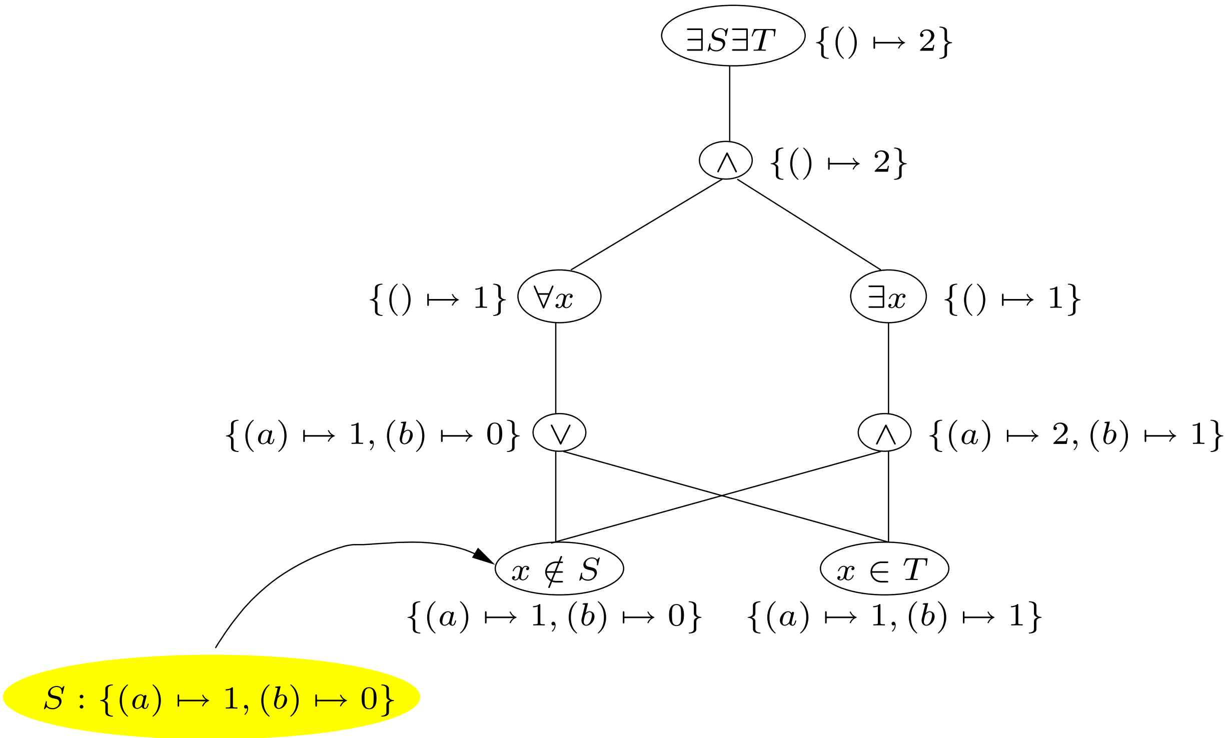
$$conflict(\phi \vee \psi)(S, k) = \max\{0\} \cup \{penalty(\phi \vee \psi)(k) - (penalty(\gamma)(k) - conflict(\gamma)(S, k)) \mid \gamma \in \{\phi, \psi\} \wedge S \in vars(\gamma)\}$$

$$conflict(|S| \leq c)(S, k) = penalty(|S| \leq c)(k)$$

$$conflict(x \in S)(S, k) = penalty(x \in S)(k)$$

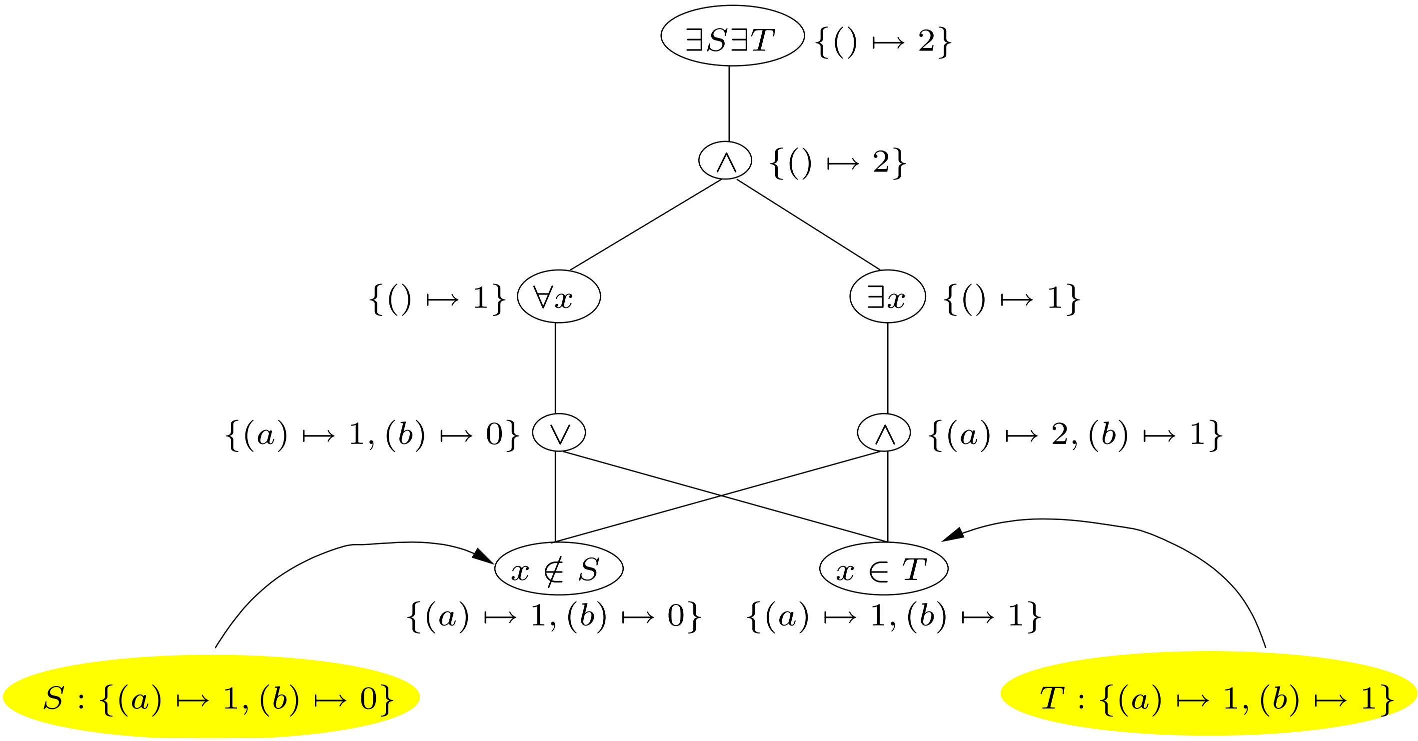
# Penalty and Conflict Tree of $S \subset T$

$$\mathcal{U} = \{a, b\},\ k(S) = \{a\},\ k(T) = \emptyset$$



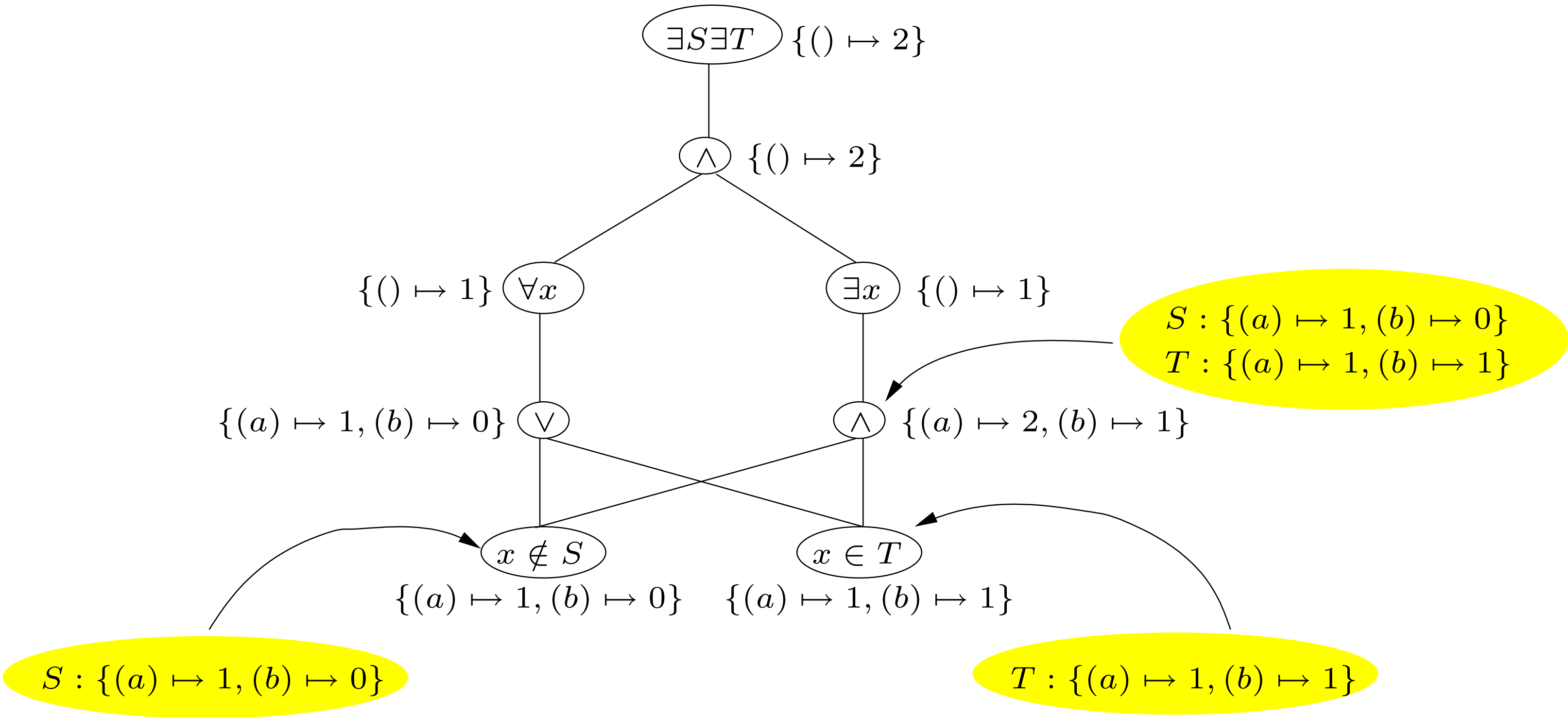
# Penalty and Conflict Tree of $S \subset T$

$$\mathcal{U} = \{a, b\},\ k(S) = \{a\},\ k(T) = \emptyset$$



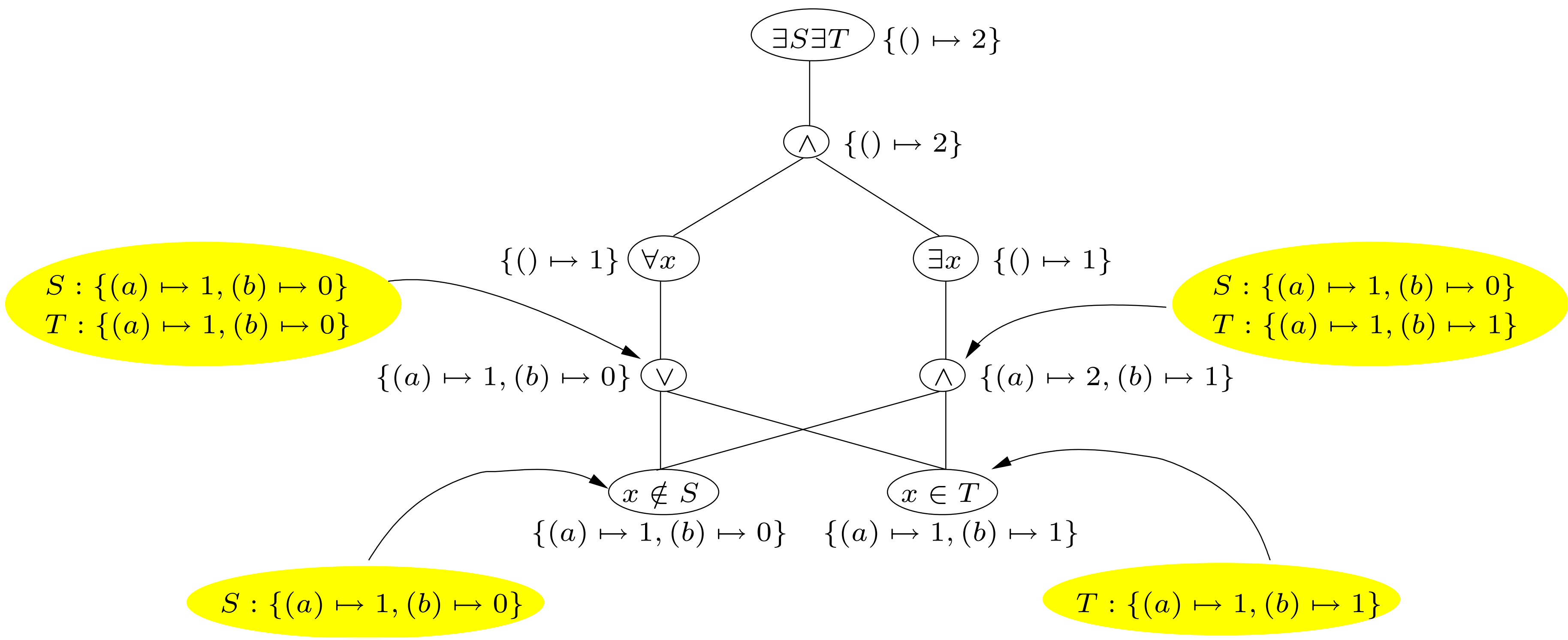
# Penalty and Conflict Tree of $S \subset T$

$$\mathcal{U} = \{a, b\},\ k(S) = \{a\},\ k(T) = \emptyset$$



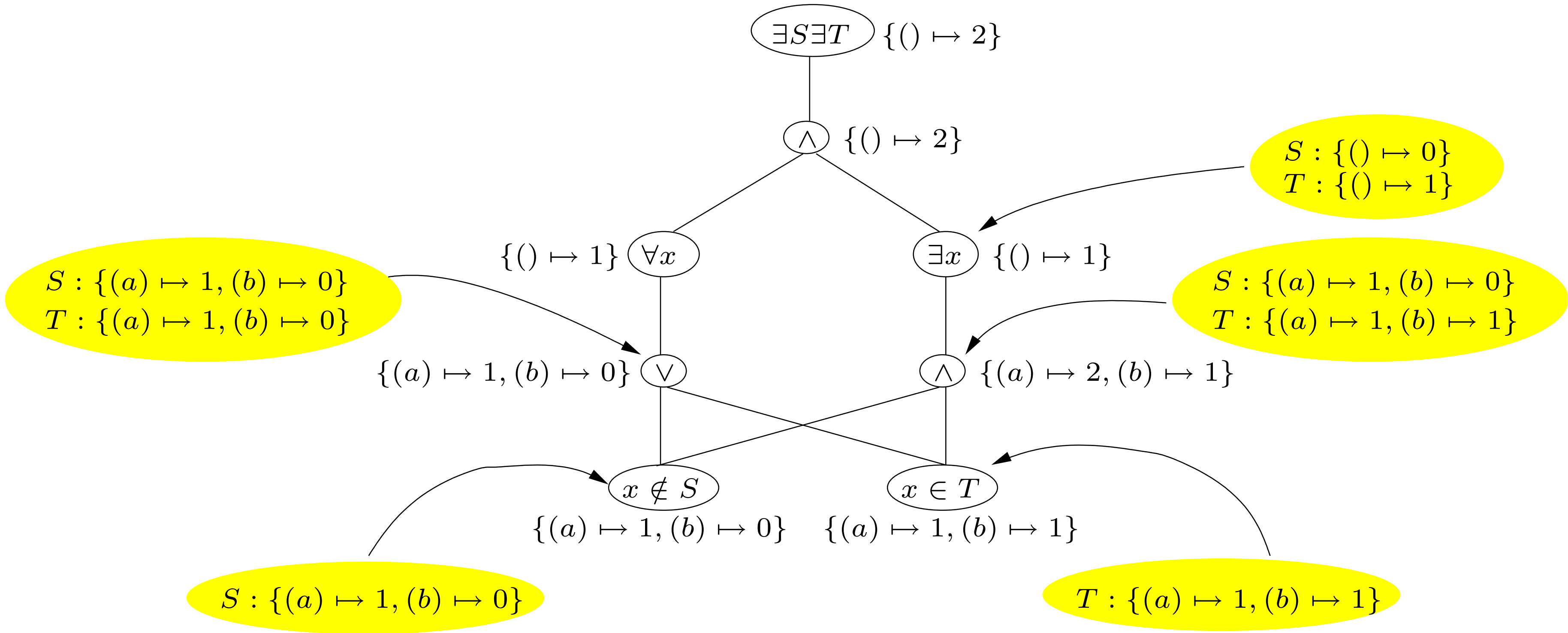
# Penalty and Conflict Tree of $S \subset T$

$$\mathcal{U} = \{a, b\},\ k(S) = \{a\},\ k(T) = \emptyset$$



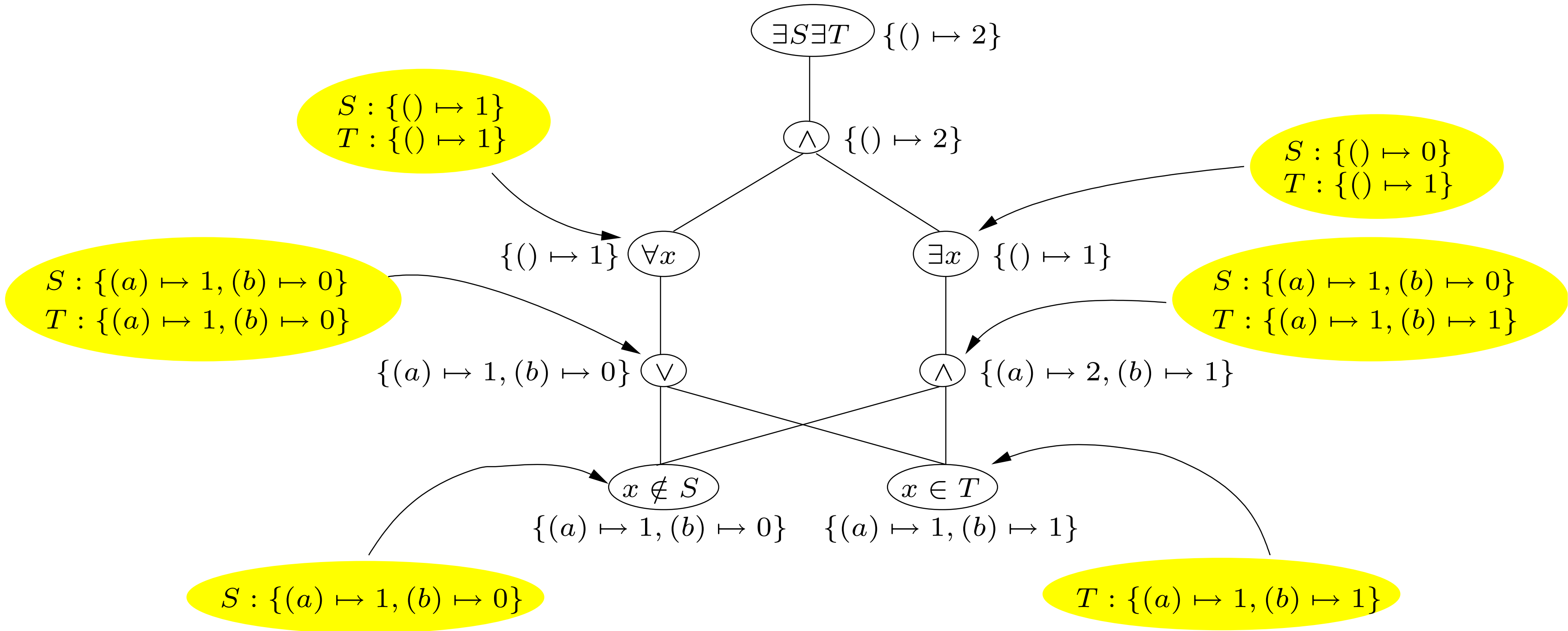
# Penalty and Conflict Tree of $S \subset T$

$$\mathcal{U} = \{a, b\},\ k(S) = \{a\},\ k(T) = \emptyset$$



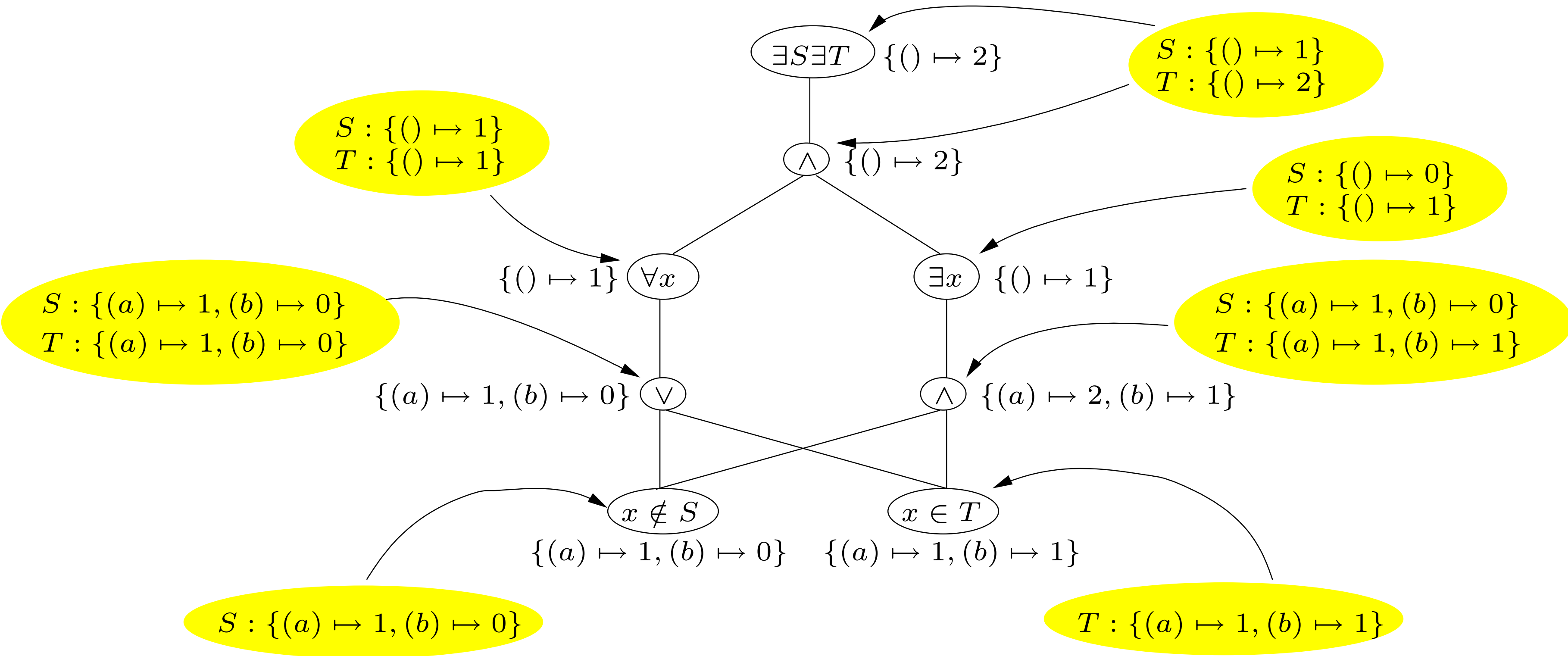
# Penalty and Conflict Tree of $S \subset T$

$$\mathcal{U} = \{a, b\},\ k(S) = \{a\},\ k(T) = \emptyset$$



# Penalty and Conflict Tree of $S \subset T$

$$\mathcal{U} = \{a, b\},\ k(S) = \{a\},\ k(T) = \emptyset$$



# Abstract Conflict of a Variable

Let $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ be a CSP, let $c \in \mathcal{C}$, and let $k$ be a configuration for $\mathcal{X}$

**Informally:** The abstract conflict of a variable $x$ with respect to $c$ and $k$ is the maximum possible penalty decrease of $c$ by only changing $k(x)$.

**Formally:** The abstract conflict function of $c$ is a function $ac(c) : \mathcal{X} \times \mathcal{K} \to \mathbb{N}$ such that:

$$ac(c)(x, k) = \max\{penalty(c)(k) - penalty(c)(\ell) \mid \ell \in \mathcal{N}_x(k)\}$$

where $\mathcal{N}_x(k)$ is the set of configurations reachable from $k$ by only changing $k(x)$.

# Properties of $\mathit{conflict}(\mathcal{F})$

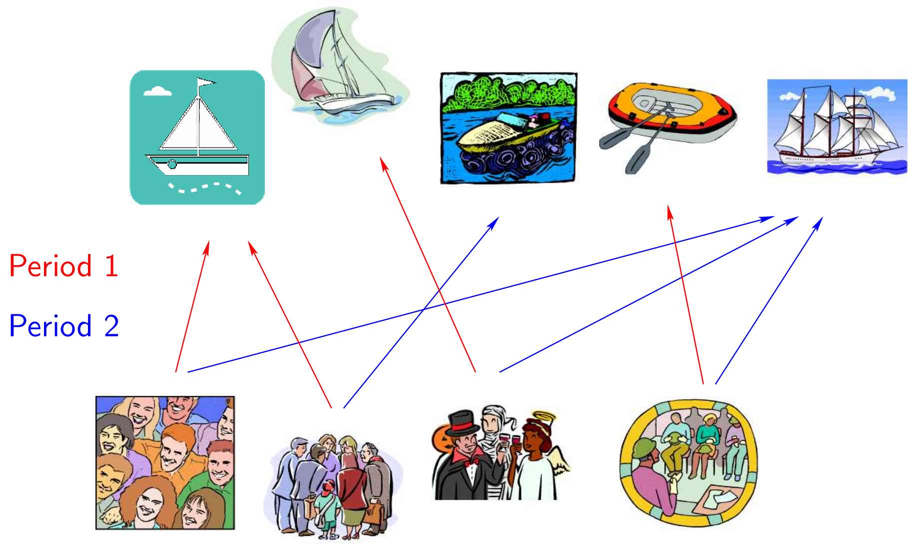
**Proposition 1.** Let $c$ be a constraint. Then $ac(c)$ is a conflict function.

**Proposition 2.** Let $\mathcal{F} \in \exists\mathrm{SOL}^+$, let $k$ be a configuration for $\mathit{vars}(\mathcal{F})$, and let $S \in \mathit{vars}(\mathcal{F})$. Then $ac(\mathcal{F})(S, k) \leq \mathit{conflict}(\mathcal{F})(S, k)$.

**Proposition 3.** Let $\mathcal{F} \in \exists\mathrm{SOL}^+$, let $k$ be a configuration for $\mathit{vars}(\mathcal{F})$, and let $S \in \mathit{vars}(\mathcal{F})$. Then $\mathit{conflict}(\mathcal{F})(S, k) \leq \mathit{penalty}(\mathcal{F})(k)$.

**Corollary.** Let $\mathcal{F} \in \exists\mathrm{SOL}^+$. $\mathit{conflict}(\mathcal{F})$ is a conflict function.

# Progressive Party Problem

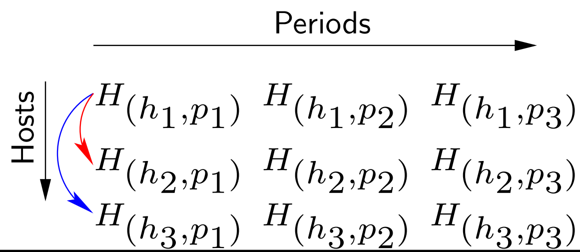



**Constraints:**
$(c_1)$ : Each guest crew shall party in each period,
$(c_2)$ : the capacity of the host boats is not exceeded,
$(c_3)$ : a guest crew visits a host boat at most once,
$(c_4)$ : two different guest crews meet at most once.

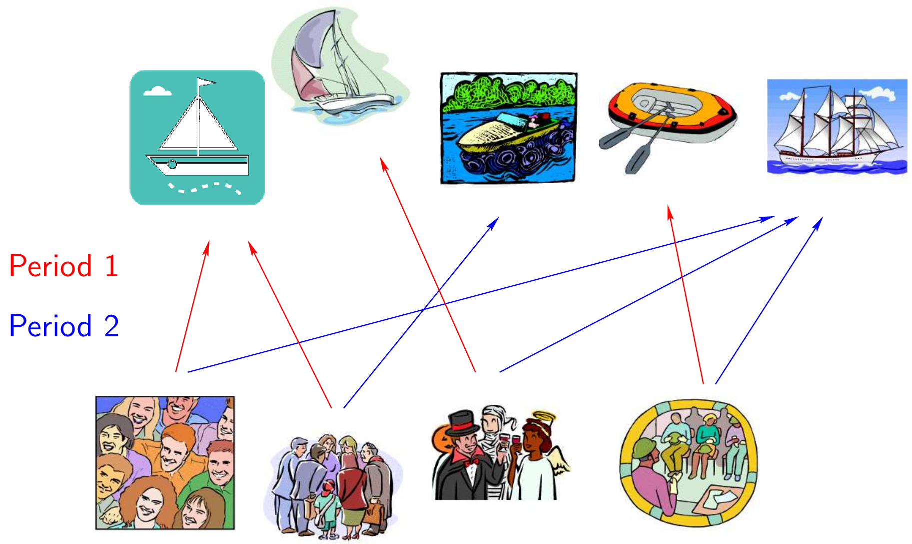
**Model:**
$P$: party periods, $H$: host boats, $G$: guest crews
$H_{(h,p)}$: set of guest boats on host boat $h$ in period $p$
$size(g)$: size of guest crew $g$
$capacity(h)$: spare capacity of host boat $h$

$(c_1) : \forall p \in P : Partition(\{H_{(h,p)} \mid h \in H\}, G)$
$(c_2) : \forall h \in H : \forall p \in P :$
$\quad MaxWeightedSum(H_{(h,p)}, size, capacity(h))$
$(c_3) : \forall h \in H : AllDisjoint(\{H_{(h,p)} \mid p \in P\})$
$(c_4) : MaxIntersect(\{H_{(h,p)} \mid h \in H \ \& \ p \in P\}, 1)$

**Neighbourhood:** Move a guest crew from a host boat $h$ to another host boat $h'$ in the same period:



# Progressive Party Problem

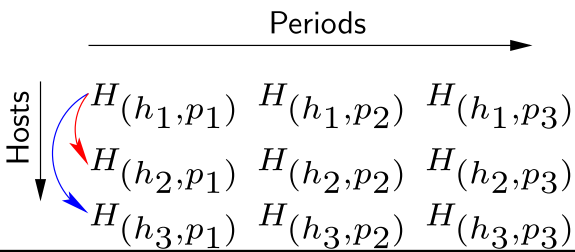



**Constraints:**
$(c_1)$ : Each guest crew shall party in each period,
$(c_2)$ : the capacity of the host boats is not exceeded,
$(c_3)$ : a guest crew visits a host boat at most once,
$(c_4)$ : two different guest crews meet at most once.

**Model:**
$P$: party periods, $H$: host boats, $G$: guest crews
$H_{(h,p)}$: set of guest boats on host boat $h$ in period $p$
$size(g)$: size of guest crew $g$
$capacity(h)$: spare capacity of host boat $h$

~~$(c_1) : \forall p \in P : Partition(\{H_{(h,p)} \mid h \in H\}, G)$~~
$(c_2) : \forall h \in H : \forall p \in P :$
$\quad MaxWeightedSum(H_{(h,p)}, size, capacity(h))$
$(c_3) : \forall h \in H : AllDisjoint(\{H_{(h,p)} \mid p \in P\})$
$(c_4) : MaxIntersect(\{H_{(h,p)} \mid h \in H \ \& \ p \in P\}, 1)$

**Neighbourhood:** Move a guest crew from a host boat $h$ to another host boat $h'$ in the same period:



# Results

Results with modelled *AllDisjoint* constraint.

| $H$/periods (fails) | 6 | 7 | | 8 | 9 | | 10 |
|---|---|---|---|---|---|---|---|
| 1-12,16 | | | | 1.3 | 3.5 | | 42.0 |
| 1-13 | | | | 16.5 | 239.3 | | |
| 1,3-13,19 | | | | 18.9 | 273.2 | (3) | |
| 3-13,25,26 | | | | 36.5 | 405.5 | (16) | |
| 1-11,19,21 | 19.8 | 186.7 | | | | | |
| 1-9,16-19 | 32.2 | 320.0 | (12) | | | | |

Results with built-in *AllDisjoint* constraint.

| $H$/periods (fails) | 6 | 7 | | 8 | 9 | | 10 |
|---|---|---|---|---|---|---|---|
| 1-12,16 | | | | 1.2 | 2.3 | | 21.0 |
| 1-13 | | | | 7.0 | 90.5 | | |
| 1,3-13,19 | | | | 7.2 | 128.4 | (4) | |
| 3-13,25,26 | | | | 13.9 | 170.0 | (17) | |
| 1-11,19,21 | 10.3 | 83.0 | (1) | | | | |
| 1-9,16-19 | 18.2 | 160.6 | (22) | | | | |

# Conclusion

## Contributions

- We use existential second-order logic with counting ($\exists\mathrm{SOL}^+$) for user-defined set constraints.
- We introduced penalty and conflict definitions for constraints modelled in $\exists\mathrm{SOL}^+$.
- We developed algorithms for incrementally maintaining the penalty and conflicts of a formula in $\exists\mathrm{SOL}^+$.

# Conclusion

## Contributions

- We use existential second-order logic with counting ($\exists\mathrm{SOL}^+$) for user-defined set constraints.
- We introduced penalty and conflict definitions for constraints modelled in $\exists\mathrm{SOL}^+$.
- We developed algorithms for incrementally maintaining the penalty and conflicts of a formula in $\exists\mathrm{SOL}^+$.

## Future Work

Revising the current local search system:

- More built-in set constraints.
- Constraints on set and integer variables, e.g., $|S| = x$.
- More efficient incremental algorithms.
- Bounded quantification in $\exists\mathrm{SOL}^+$, such as $\forall(x \in S)\phi(x)$