SAT Problems

Steffen Hölldobler and Norbert Manthey
International Center for Computational Logic
Technische Universität Dresden
Germany

- Propositional Logic
- Semantics
- Propositional SAT Problems
- Conjunctive Normal Form
- Resolution
- Examples

"Logic is everywhere ..."
Propositional Logic

Definition An alphabet of propositional logic consists of

- a (countably) infinite set $\mathcal{R}$ of propositional variables
- the set $\{\neg/1$, $\land/2$, $\lor/2$, $\rightarrow/2$, $\leftrightarrow/2\}$ of connectives and
- the special characters "(" and ")"

$\therefore/n$ denotes the arity of $\cdot$

Different alphabets of propositional logic differ in $\mathcal{R}$ and, hence, alphabets are usually specified by specifying $\mathcal{R}$

In this lecture, $\mathcal{R}$ is usually $\mathbb{N}^+$
Propositional Formulas

Definition An atomic formula, briefly called atom, is a propositional variable

Definition The set of propositional formulas is the smallest set $\mathcal{L}(\mathcal{R})$ of strings over an alphabet $\mathcal{R}$ of propositional logic with the following properties:

1. If $F$ is an atomic formula then $F \in \mathcal{L}(\mathcal{R})$
2. If $F \in \mathcal{L}(\mathcal{R})$ then $\neg F \in \mathcal{L}(\mathcal{R})$
3. If $\circ/2$ is a binary connective and $F, G \in \mathcal{L}(\mathcal{R})$ then $(F \circ G) \in \mathcal{L}(\mathcal{R})$

Definition A literal is an atom or a negated atom;
The complement $\overline{L}$ of a literal $L$ is defined as follows:

- If $L$ is an atom $A$ then $\overline{L} = \neg A$
- if $L$ is a negated atom $\neg A$ then $\overline{L} = A$

A pair $L, \overline{L}$ of literals is said to be complementary
Notations and Conventions

- A (possibly indexed) denotes an atom
- L (possibly indexed) denotes a literal
- F, G, H (possibly indexed) denote propositional formulas
- F, G, H denote sets of propositional formulas

- It is sometimes convenient to write \(-n\) instead of \(\neg n\), where \(n \in \mathbb{N^+}\)

- Let \(S\) be a set of literals

  \[\overline{S} = \{\overline{L} \mid L \in S\}\]

  \(\overline{S}\) is sometimes called the complement of \(S\)
Semantics

- The set of truth values is the set $\{\top, \bot\}$
- We consider the following functions on $\{\top, \bot\}$:
  - Negation $\neg^*$ / 1
  - Conjunction $\land^*$ / 2
  - Disjunction $\lor^*$ / 2
  - Implication $\to^*$ / 2
  - Equivalence $\leftrightarrow^*$ / 2

<table>
<thead>
<tr>
<th></th>
<th>$\neg^*$</th>
<th>$\land^*$</th>
<th>$\lor^*$</th>
<th>$\to^*$</th>
<th>$\leftrightarrow^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\top$</td>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\top$</td>
<td>$\top$</td>
<td>$\top$</td>
</tr>
<tr>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\bot$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\top$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\bot$</td>
</tr>
</tbody>
</table>
Interpretations

- **Definition** An interpretation $I$ consists of the set $\{\top, \bot\}$ and a mapping $\cdot^I : \mathcal{L}(\mathcal{R}) \to \{\top, \bot\}$ with:

  $$[F]^I = \begin{cases} 
  \neg^*[G]^I & \text{if } F \text{ is of the form } \neg G \\
  ([G_1]^I \circ^* [G_2]^I) & \text{if } F \text{ is of the form } (G_1 \circ G_2)
  \end{cases}$$

- **Given** $F \in \mathcal{L}(\mathcal{R})$

- **Let** $\mathcal{R}_F = \{A \in \mathcal{R} \mid A \text{ occurs in } F\}$ and $n = |\mathcal{R}_F|$

- **Definition** Two interpretations $I$ and $J$ are equal for $F$, in symbols $I \simeq_F J$, iff for all $A \in \mathcal{R}_F$ we find $A^I = A^J$

- **Proposition** $\simeq_F$ is an equivalence relation defining $2^n$ different equivalence classes on the set of all interpretations of $\mathcal{L}(\mathcal{R})$

- For each of the equivalence classes defined by $\simeq_F$ we can choose as representative the interpretation $I$ with $A^I = \bot$ for all $A \in \mathcal{R} \setminus \mathcal{R}_F$

- Such an interpretation $I$ is called an interpretation for $F$

- The set of interpretations for $F$ is finite; its cardinality is $2^n$
Models

► Definition  An interpretation $I$ for $F$ is called model for $F$ ($I \models F$) iff $[F]^I = \top$

► Definition  

$F$ is satisfiable  iff  there is a model for $F$
$F$ is unsatisfiable  iff  there is no model for $F$
$F$ is valid  iff  all interpretations for $F$ are models for $F$
$F$ is falsifiable  iff  some interpretation for $F$ is not a model for $F$

► Definition  An interpretation $I$ is called model for a set $\mathcal{G}$ of formulas ($I \models \mathcal{G}$) iff $I$ is a model for all $F \in \mathcal{G}$

► The notions of satisfiability, unsatisfiability, validity and falsifiability can be extended to sets of formulas in the obvious way
Representation of Interpretations

- An interpretation \( I \) for \( F \) is uniquely defined by specifying how \( I \) acts on \( \mathcal{R}_F \)
  - \( I \) can be represented by a sequence \( \hat{I} \) of literals from \( \mathcal{R}_F \cup \overline{\mathcal{R}_F} \) such that \( L \in \hat{I} \) iff \( L^I = \top \)

- Note
  - \( \hat{I} \) is a mapping
    - \( \hat{I} \) does not contain a complementary pair of literals
  - \( \hat{I} \) is a total mapping
    - For each \( A \in \mathcal{R}_F \) either \( A \in \hat{I} \) or \( \overline{A} \in \hat{I} \) but not both
  - In the sequel, we will identify \( I \) with \( \hat{I} \).

- Definition  Let \( J \) be a sequence of literals from \( \mathcal{R}_F \cup \overline{\mathcal{R}_F} \) such that \( J \) does not contain a complementary pair; \( J \) is a partial interpretation for \( F \) iff there is an \( A \in \mathcal{R}_F \) such that neither \( A \in J \) nor \( \overline{A} \in J \).
Some Additional Notations and Conventions

- $I$ and $J$ (possibly indexed) denote (partial) interpretations.
- We often write $F^I$ instead of $[F]^I$.
- We define the following precedence hierarchy among connectives:
  \[ \neg \succ \{ \lor, \land \} \succ \rightarrow \succ \leftrightarrow \]
- We sometimes omit parentheses taking into account that conjunction and disjunction are associative and commutative.
- Let $J$ be a (partial) interpretation for $F$ and $C$ a disjunction of literals:
  - $J$ satisfies $C$ ($J \models C$) iff $J$ contains a literal occurring as disjunct in $C$.
  - $J$ falsifies $C$ ($J \not\models C$) iff for each disjunct $L$ of $C$ we find $\overline{L} \in J$.
- Let $J$ be a sequence of literals; It it sometimes convenient to represent $J$ in the form $I', L, I$, where $L$ is a literal occurring in $J$ and $I', I$ are the subsequences occurring in $J$ before and after $L$, respectively.
Definition A propositional satisfiability problem, briefly called SAT, consists of a formula $F \in \mathcal{L}(\mathcal{R})$, and is the problem to decide whether $F$ is satisfiable.

SAT is a combinatorial decision problem

- **Decision variant** yes/no answer
- **Search variant** find a model if $F$ is satisfiable
- **All models variant** find all models if $F$ is satisfiable
A Simple SAT Instance

Let $F = 1$

\[\land (1 \lor 2)\]
\[\land (1 \rightarrow 3)\]
\[\land (1 \land 3 \rightarrow 4)\]
\[\land (5 \lor 6)\]
\[\land (5 \rightarrow 7)\]
\[\land (5 \lor 8)\]
\[\land (\overline{7} \lor 8)\]

$(1, 2, 3, 4, 5, 6, 7, 8)$ is a model for $F$

Hence, $F$ is satisfiable

How can we find such a model?
Model Finding – First Ideas

- Reconsider $F = 1$

  $C_1$ 
  $\land (1 \lor 2)$ 
  $\land (1 \rightarrow 3)$ 
  $\land (1 \land 3 \rightarrow 4)$ 
  $\land (5 \lor 6)$ 
  $\land (5 \rightarrow 7)$ 
  $\land (\overline{5} \lor 8)$ 
  $\land (\overline{7} \lor 8)$

  $C_2$ 
  $\land (1 \lor 2)$ 

  $C_3$ 
  $\land (1 \rightarrow 3)$ 

  $C_4$ 
  $\land (1 \land 3 \rightarrow 4)$ 

  $C_5$ 
  $\land (5 \lor 6)$ 

  $C_6$ 
  $\land (5 \rightarrow 7)$ 

  $C_7$ 
  $\land (\overline{5} \lor 8)$ 

  $C_8$ 
  $\land (\overline{7} \lor 8)$

- Idea

  Initialize $J := ()$
  and add literals to $J$
  such that $J \models C_i$
  for all $1 \leq i \leq 8$

- ▶ Because $C_1$ we set $J := (1)$ and thus $J \models C_1$.
- ▶ Because $1 \in J$ we find $J \models C_2$.
- ▶ Because $1 \in J$ and $C_3$ we set $J := (1, 3)$ and thus $J \models C_3$
- ▶ Because $1, 3 \in J$ and $C_4$ we set $J := (1, 3, 4)$, and thus $J \models C_4$
- ▶ None of $C_5 - C_8$ forces the addition of a literal; we choose $J := (1, 3, 4, 5)$
- ▶ Because $5 \in J$ we find $J \models C_5$
- ▶ Because $5 \in J$ and $C_6$ we set $J := (1.3.4, 5, 7)$, and thus $J \models C_6$
- ▶ Because $5 \in J$ and $C_7$ we set $J := (1, 3, 4, 5, 7, 8)$ and thus $J \models C_7$
- ▶ Because $7, 8 \in J$ we find $J \not\models C_8$; we have a conflict
Model Finding – First Ideas Continued

▶ Reconsider \( F = 1 \)

\[
\begin{align*}
C_1 & \quad \land (1 \lor 2) \\
C_2 & \quad \land (1 \rightarrow 3) \\
C_3 & \quad \land (1 \land 3 \rightarrow 4) \\
C_4 & \quad \land (5 \lor 6) \\
C_5 & \quad \land (5 \rightarrow 7) \\
C_6 & \quad \land (5 \lor 8) \\
C_7 & \quad \land (7 \lor 8).
\end{align*}
\]

▶ Recall \( J := (1, 3, 4, 5, 7, 8) \) has led to a conflict

▶ We backtrack and set \( J := (1, 3, 4, 5) \)

▶ Because \( 5 \in J \) and \( C_5 \) we set \( J := (1, 3, 4, 5, 6) \) and thus \( J \models C_5 \)

▶ Because \( 5 \in J \) we find \( J \models C_6 \) and \( J \models C_7 \)

▶ In order to satisfy \( C_8 \) we choose \( J := (1, 3, 4, 5, 6, 7) \) and thus \( J \models C_8 \)

▶ \( J \) is turned into a total interpretation by adding \( 2, 8 \);
the choice was arbitrary; we could have added \( 2, 8 \) or \( 2, 8 \) or \( 2, 8 \)
Remarks and Notational Conventions

- Literals marked with a dot are called decision literals; all others are called propagation literals.

- If $J$ is a partial interpretation, then $J, L$ is the interpretation obtained by adding $L$ to $J$.

  Note: $J, L$ may be total.
Decision Levels

- Partial interpretations will sometimes be written in the form 
  \[ P_0, \dot{L}_1, P_1, \ldots, \dot{L}_k, P_k, \]
  where \( P_i, 1 \leq i \leq k \), are sequences of propagation literals

- The decision literals partition the elements of the interpretation into decision levels

- Literals occurring in \( L_i, P_i \) are assigned decision level \( i \)

- Likewise,
  \[ J, \dot{L}, P \]
  denotes a partial interpretation, where

- \( J \) is a partial interpretation

- \( \dot{L} \) is decision literal and

- \( P \) is a sequence of propagation literals

Note that \( \dot{L} \) is the decision literal with the highest level in \( J, \dot{L}, P \)
Subformulas

**Definition** Let $F$ be a propositional formula; The set of subformulas of $F$ is the smallest set of formulas $S(F)$ satisfying the following conditions:

1. $F \in S(F)$
2. If $\neg G \in S(F)$, then $G \in S(F)$
3. If $G_1 \circ G_2 \in S(F)$, then $G_1, G_2 \in S(F)$

**Example**

$$S(\neg ((p_1 \rightarrow p_2) \lor p_1)) = \{\neg ((p_1 \rightarrow p_2) \lor p_1), ((p_1 \rightarrow p_2) \lor p_1), (p_1 \rightarrow p_2), p_1, p_2\}$$
Semantic Equivalence

**Definition**  Two propositional formulas $F$ and $G$ are **semantically equivalent**, in symbols $F \equiv G$, iff for all interpretations $I$ we have: $I \models F$ iff $I \models G$

**Theorem**  Some equivalence laws:

- $\neg \neg F \equiv F$  \hspace{1cm} \text{double negation}
- $\neg (F \land G) \equiv \neg F \lor \neg G$
- $\neg (F \lor G) \equiv \neg F \land \neg G$  \hspace{1cm} \text{de Morgan}
- $F \land (G \lor H) \equiv (F \land G) \lor (F \land H)$
- $F \lor (G \land H) \equiv (F \lor G) \land (F \lor H)$  \hspace{1cm} \text{distributivity}
- $F \leftrightarrow G \equiv (F \land G) \lor (\neg G \land \neg F)$  \hspace{1cm} \text{equivalence}
- $F \rightarrow G \equiv \neg F \lor G$  \hspace{1cm} \text{implication}
- $F \lor G \equiv F$, if $F$ is valid
- $F \land G \equiv G$, if $F$ is valid  \hspace{1cm} \text{tautology}
- $F \lor G \equiv G$, if $F$ is unsatisfiable
- $F \land G \equiv F$, if $F$ is unsatisfiable  \hspace{1cm} \text{unsatisfiability}
Replacement

- $F[G \leftrightarrow H]$ denotes the formula obtained from $F$ by replacing an occurrence of $G \in S(F)$ by $H$
  - Usually, the context determines which occurrence is meant
  - Sometimes the condition $G \in S(F)$ is omitted
    In this case, if $G \not\in S(F)$, then $F[G \leftrightarrow H] = F$
- **Replacement Theorem**  If $G \equiv H$ then $F[G \leftrightarrow H] \equiv F$
Generalized Disjunctions and Conjunctions

- Generalized disjunction \([F_1, \ldots, F_n] := F_1 \lor \ldots \lor F_n\)

- Generalized conjunction \(\langle F_1, \ldots, F_n \rangle := F_1 \land \ldots \land F_n\)

- Empty generalized disjunction \([\ ]\) with \([\ ]^I = \bot\) for all \(I\)

- Empty generalized conjunction \(\langle \rangle\) with \(\langle \rangle^I = \top\) for all \(I\)

- Note \(n \land \overline{n}\) is unsatisfiable, whereas \(n \lor \overline{n}\) is valid, where \(n \in \mathbb{N}^+\)

- Notation We consider \(\langle \rangle\) and \([\ ]\) as abbreviations for \(1 \lor \overline{1}\) and \(1 \land \overline{1}\), resp.
Clauses and Conjunctive Normal Forms

Definition

- A clause is a generalized disjunction \([L_1, \ldots, L_n]\), \(n \geq 0\), where every \(L_i, 1 \leq i \leq n\), is a literal
- A clause is a Horn clause if at most one disjunct is an atom
- A clause is a unit clause if it contains precisely one literal
- A clause is a binary clause if it contains precisely two literals

Definition

- A formula is in conjunctive normal form (clause form, CNF) iff it is of the form \(\langle C_1, \ldots, C_m\rangle\), \(m \geq 0\), and every \(C_j, 1 \leq j \leq m\), is a clause
- A formula \(F\) in CNF is a Horn formula if it contains only Horn clauses
- A formula \(F\) in CNF is said to be in \(n\)-CNF if each clause occurring in \(F\) has at most \(n\) literals
More Notations and Conventions

► *C* (possibly indexed) denotes a clause

► *C, L* and *F, C* denote *C ∨ L* and *F ∧ C*, respectively, where *C* is a clause and *F* a CNF-formula

► Clauses and CNF-formulas are sometimes considered as sets of literals and clauses, respectively, in which case
  ▶ *L*, 1 ≤ *i* ≤ *n*, are said to be elements of [*L*₁, . . . , *L*ₙ] and
  ▶ *C*, 1 ≤ *j* ≤ *m*, are said to be elements of ⟨*C*₁, . . . , *C*ₘ⟩

Note that in sets duplicates are removed!

► It should be clear from the context whether clauses and CNF-formulas are considered as sets or generalized disjunctions and conjunctions, respectively

► When writing *C = C′, L*
  we do not suppose that *L* is the “last” literal occurring in *C*
  but some literal occurring in *C*
  and *C′* is the disjunction or set of the “remaining” literals occurring in *C*

► A similar convention applies to *F = F′, C*
The Function \textit{lits}

- Let \textit{lits} be the following function from the set of clauses to the set of literals

\[
lits(C) = \begin{cases} 
\emptyset & \text{if } C = [] \\
lits(C') \cup \{L\} & \text{if } C = C', L 
\end{cases}
\]

- It is extended to a function from the set of CNF-formulas to the set of literals

\[
lits(F) = \begin{cases} 
\emptyset & \text{if } F = \langle \rangle \\
lits(F') \cup lits(C) & \text{if } F = F', C 
\end{cases}
\]
The Function atoms

Let atoms be the following function from the set of literals to the set of atoms

$$\text{atoms}(L) = \begin{cases} \{A\} & \text{if } L = A \\ \{A\} & \text{if } L = \neg A \end{cases}$$

It is extended to a function from the set of clauses to the set of atoms

$$\text{atoms}(C) = \begin{cases} \emptyset & \text{if } C = [] \\ \text{atoms}(C') \cup \text{atoms}(L) & \text{if } C = C', L \end{cases}$$

It is extended to a function from the set of CNF-formulas to the set of atoms

$$\text{atoms}(F) = \begin{cases} \emptyset & \text{if } F = \langle \rangle \\ \text{atoms}(F') \cup \text{atoms}(C) & \text{if } F = F', C \end{cases}$$
Transformation into Clause Form

- **Theorem**  There is an algorithm which transforms any propositional formula into a semantically equivalent formula in clause form.

- **Observation**
  - All equivalences can be eliminated using the law:
    \[
    F \leftrightarrow G \equiv (F \land G) \lor (\neg F \land \neg G)
    \]
    - \(F\) and \(G\) are copied which may lead to a combinatorial explosion!
    - Construct a sequence of examples demonstrating this explosion.

  - All implications can be eliminated using the law:
    \[
    F \rightarrow G \equiv \neg F \lor G
    \]
    - Hence, we assume that only the connectives \(\neg\), \(\land\) and \(\lor\) occur in formulas.
An Algorithm for the Transformation into Clause Form

Input: A propositional formula $F$
Output: A formula, which is in conjunctive normal form and is equivalent to $F$

$G := \langle[F] \rangle$ (G is a conjunction of disjunctions)

While $G$ is not in conjunctive normal form do:
   Select a non-clausal element $H$ from $G$
   Select a non-literal element $K$ from $H$
   Apply the rule among the following ones which is applicable

   $\neg\neg D \quad (D_1 \land D_2) \quad \neg(D_1 \land D_2) \quad (D_1 \lor D_2) \quad \neg(D_1 \lor D_2)$

   $D \quad D_1 \mid D_2 \quad \neg D_1, \neg D_2 \quad D_1, D_2 \quad \neg D_1 \mid \neg D_2$

A rule $\frac{D}{D'}$ is applicable to $K$ if $K$ is of the form $D$
If applied, then $K$ is replaced by $D'$

A rule $\frac{D}{D_1 \mid D_2}$ is applicable to $K$ if $K$ is of the form $D$
If applied, $H$ is replaced by two disjunctions
The first one is obtained from $H$ by replacing the occurrence of $D$ by $D_1$
The second one is obtained from $H$ by replacing the occurrence of $D$ by $D_2$
An Example

- Let $F = p \land (p \rightarrow q) \rightarrow q$
- $F$ is valid
- Eliminating implications yields
  $$\lnot(p \land (\lnot p \lor q)) \lor q$$
- Applying the algorithm yields
  $${\langle \lnot(p \land (\lnot p \lor q)) \lor q \rangle, \langle \lnot(p \land (\lnot p \lor q), q \rangle, \langle [\lnot p, \lnot(\lnot p \lor q), q] \rangle, \langle [\lnot p, \lnot p \land \lnot q, q \rangle, \langle [\lnot p, \lnot p, q], [\lnot p, \lnot q, q] \rangle, \langle [\lnot p, p, q], [\lnot p, \lnot q, q] \rangle}$$
- Both clauses in the final formula contain a complementary pair of literals
Remarks

- An application of a rule of the form $\frac{D}{D_1 | D_2}$ may lead to copies of subformulas
  - May this lead to a combinatorial explosion?
  - If this is the case,
    then construct a sequence of examples showing the explosion
  - If this is not the case, then prove it
Definitional Transformation

- The size of a formula may grow exponentially during normalization
- Can we do better?
  - Unfortunately, the shortest CNF of some $F$ is exponential in the size of $F$
  - Luckily, we may use a weaker concept
- **Definitional transformation**  
  Tseitin: On the complexity of derivation in propositional calculus. Leningrad Seminar on Mathematical Logic, 1970
  - Let $F$ be a formula, $G \in S(F)$ and $p \notin S(F)$ a propositional variable
  - Replace $F$ by $F[G \leftrightarrow p] \land (p \leftrightarrow G)$
- **Some observations**
  - $F \not\equiv F[G \leftrightarrow p] \land (p \leftrightarrow G)$
  - $F$ is satisfiable iff $F[G \leftrightarrow p] \land (p \leftrightarrow G)$ is satisfiable (equi-satisfiable)
  - The previously mentioned exponential growth can be avoided
Reduct of a CNF-Formula

**Definition** Let $F$ be a CNF-formula and $J$ a partial interpretation. The reduct of $F$ wrt $J$ ($F|_J$) is obtained by applying the following transformations to $F$: For all $L \in J$ do

- Remove all clauses in $F$ which contain $L$
- Remove all occurrences of $\overline{L}$

Let $F$ be the following formula:

$$\langle[1], [1, 2], [\overline{1}, 3], [\overline{1}, 3, 4], [5, 6], [\overline{5}, 7], [\overline{5}, 8], [\overline{7}, \overline{8}]\rangle$$

Then,

- $F|_{(1)} = \langle[3], [\overline{3}, 4], [5, 6], [\overline{5}, 7], [\overline{5}, 8], [\overline{7}, \overline{8}]\rangle$
- $F|_{(1,3)} = \langle[4], [5, 6], [\overline{5}, 7], [\overline{5}, 8], [\overline{7}, \overline{8}]\rangle$
- $F|_{(1,3,4)} = \langle[5, 6], [\overline{5}, 7], [\overline{5}, 8], [\overline{7}, \overline{8}]\rangle$
- $F|_{(1,3,4,5)} = \langle[6], [\overline{7}, \overline{8}]\rangle$
- $F|_{(1,3,4,5,6)} = \langle[\overline{7}, \overline{8}]\rangle$
- $F|_{(1,3,4,\overline{5},6,\overline{7})} = \langle\rangle$
Reduct of a Clause

Definition Let $C$ be a clause and $J$ be a (partial or total) interpretation. The reduct of $C$ wrt $J$, in symbols $C|_J$, is

- $\langle \rangle$ if $C \cap J \neq \emptyset$
- the clause obtained from $C$ by removing all occurrences of $\overline{L}$ for all $L \in J$
Conflicts

Definition Let $F$ be a CNF-formula and $J$ a (partial or total) interpretation for $F$

- $J$ satisfies $F$ (in symbols, $J \models F$) iff $F|_J$ is empty
- $J$ falsifies $F$ (in symbols, $J \not\models F$) iff $F|_J$ contains the empty clause;
  In this case, $J$ is sometimes called conflict for $F$
Propositional Resolution

- In the following clauses are considered to be sets

- **Definition**  Let $C_1$ be a clause containing $L$ and $C_2$ be a clause containing $\bar{L}$; The (propositional) resolvent of $C_1$ and $C_2$ with respect to $L$ is the clause

$$ (C_1 \setminus \{L\}) \cup (C_2 \setminus \{\bar{L}\}) $$

$C$ is said to be a **resolvent of** $C_1$ and $C_2$ iff there exists a literal $L$ such that $C$ is the resolvent of $C_1$ and $C_2$ wrt $L$
Linear Resolution Derivations

Definition: Let $C$, $D$ be clauses and $\mathcal{F}$ a set of formulas.

- A linear resolution derivation from $C$ wrt $\mathcal{F}$
  is a sequence $(D_i \mid i \geq 0)$ of clauses such that
  - $D_0 = C$ and
  - $D_i$ is a resolvent of $D_{i-1}$ and some $E \in \mathcal{F}$ for all $i > 0$

- A linear resolution derivation from $C$ to $D$ wrt $\mathcal{F}$ is
  - a finite linear resolution derivation $(D_i \mid 0 \leq i \leq n)$ from $C$ wrt $\mathcal{F}$
  - such that $D_n = D$
Example: Sudoku Puzzles

- Let $n \in \mathbb{N}$; A Sudoku puzzle
  - consists of an $n^2 \times n^2$ grid
  - made up of $n \times n$ subgrids called blocks
  - with some integers from $[1, n^2]$ placed in some cells
  - where some of these placements are predefined

- The problem is
  - to assign $i \in [1, n^2]$ to each cell of the grid such that
  - each row, column and block contains exactly one occurrence of each integer in $[1, n^2]$

- There are more than $6 \times 10^{12}$ 3-Sudoku puzzles

- Sudoku puzzles with $n > 3$ appear to be difficult to solve for humans
A Simple 3-Sudoku

+-------+-------+-------+
| - - 4 | 2 3 9 | - - - |
| - 8 - | 5 - 6 | - - - |
| 9 - - | 8 - 4 | - 6 - |
+-------+-------+-------+
| 5 7 1 | - - - | 9 4 6 |
| 8 - - | - - - | - - 3 |
| 2 3 9 | - - - | 7 8 1 |
+-------+-------+-------+
| - - - | 4 - 8 | - - 7 |
| - - 3 | 9 - 7 | - 1 - |
| - - - | 1 2 3 | 4 - - |
+-------+-------+-------+
A SAT Encoding of $n$-Sudokus (1)

- $(x, y, v)$ represents the fact that value $v$ is in the cell $x, y$
- **Definedness** Each cell contains one element of $[1, n^2]$

\[ \bigwedge_{x=1}^{n^2} \bigwedge_{y=1}^{n^2} \bigvee_{v=1}^{n^2} (x, y, v) \]

- **Uniqueness for Cells** Each cell has at most one value

\[ \bigwedge_{x=1}^{n^2} \bigwedge_{y=1}^{n^2} \bigwedge_{v=1}^{n^2-1} \bigwedge_{w=v+1}^{n^2} ((x, y, v) \rightarrow \neg(x, y, w)) \]

- **Uniqueness for Rows** All numbers in $[1, n^2]$ must occur in every row

\[ \bigwedge_{x=1}^{n^2} \bigwedge_{v=1}^{n^2} \bigwedge_{y=1}^{n^2-1} \bigwedge_{w=y+1}^{n^2} ((x, y, v) \rightarrow \neg(x, w, v)) \]
A SAT Encoding of $n$-Sudokus (2)

- **Uniqueness for Columns**  All numbers in $[1, n^2]$ must occur in every column

\[
\bigwedge_{y=1}^{n^2} \bigwedge_{v=1}^{n^2} \bigwedge_{x=1}^{n^2-1} \bigwedge_{w=x+1}^{n^2} ((x, y, v) \rightarrow \neg(w, y, v))
\]

- **Uniqueness for Blocks**  All numbers in $[1, n^2]$ must occur in every block

\[
\bigwedge_{i=0}^{n-1} \bigwedge_{j=0}^{n-1} \bigwedge_{x=n\cdot i+n}^{n\cdot i+n} \bigwedge_{y=n\cdot j+n}^{n\cdot j+n} \bigwedge_{v=1}^{n^2-1} \bigwedge_{w=v+1}^{n^2} ((x, y, v) \rightarrow \neg(x, y, w))
\]

- **Claim**  Let $\mathcal{F}$ be the set of formulas encoding a Sudoku puzzle
  Each model for $\mathcal{F}$ specifies a solution for the puzzle
Example: Planning

- Situation Calculus
- A Simple Planning Language
- Planning as Satisfiability Testing
- Solving Planning Problems
Situation Calculus

- Situation calculus based planning as deduction

  - General properties of causality, and certain obvious but until now unformalized facts about the possibility and results of actions, are given as axioms
  - It is a logical consequence of the facts of a situation and the general axioms that certain persons can achieve certain goals by taking certain actions
  - Block $a$ is on block $b$ after performing action $\text{move}(a, b)$ in state $s_1$
    
    $$\text{on}(a, b, \text{result}(\text{move}(a, b), s_1))$$

  - Inherently first-order
Planning as Satisfiability Testing

- We are interested only in finite plans containing no more than a given number of actions
  - A restricted approach which is equivalent to a finite propositional system
  - Planning as satisfiability testing instead of planning as deduction
    Kautz, Selman: Planning as Satisfiability.
    In: Proceedings 10th European Conference on Artificial Intelligence, 359-363: 1992
A Simple Planning Language

- We will use schemas to denote finite sets of propositional formulas.
- A schema is a function-free typed predicate logic formula with equality.
  - Two types: block and time.
  - Each type contains a finite set of individuals denoted by unique constants.
  - **Table, a, b, ...** are constants of type block.
  - The set constants of type time is a finite set of integers \([1, n]\).
  - The precedence order is extended to:
    \[
    \neg \succ \{\lor, \land\} \succ \rightarrow \succ \leftrightarrow \succ \{\forall, \exists\}
    \]
  - **X, Y, ...** denote variables of type block.
  - **T** denotes a variable of type time ranging over \([1, n - 1]\).
  - **T'** denotes a variable of type time ranging over \([1, n]\).
  - Arithmetic expressions like **T + 1** are interpreted when schemas are written in full.
Predicates

- **on**(X, Y, T) denotes that block X is on top of block Y at time T
- **clear**(X, T) denotes that block X is clear at time T
- **move**(X, Y, Z, T) denotes that X is moved from the top of Y to the top of Z between T and T + 1
- **X = Y** denotes that X and Y are the same block
Equality Constraints

- **Equalities**

\[ a = a \mid a \text{ is a constant of type block} \]

- **Inequalities**

\[ a \neq b \mid a \text{ and } b \text{ are two different constants of type block} \]
Let \([1, n]\) be the range of integers

- \(n\) states \(s_1, \ldots, s_n\)
- \(n - 1\) actions \(a_1, \ldots, a_{n-1}\) with \(a_i\) leading from \(s_i\) to \(s_{i+1}\), \(1 \leq i \leq n - 1\)

Initial conditions are formulas in which only 1 appears as term of type \textit{time}

- \(\text{on}(a, b, 1) \land \text{on}(b, \text{table}, 1) \land \text{clear}(a, 1)\)

Goal conditions are formulas in which only \(n\) appears as term of type \textit{time}

- With \(n = 3\) we may consider \(\text{on}(b, a, 3)\)
Domain Constraints

- The table is always clear \((\forall T') \text{clear}(\text{table}, T')\)
- A block except the table cannot be clear and support a block at the same time

\[(\forall X, Y, T') (Y \neq \text{table} \rightarrow \neg(\text{clear}(Y, T') \land \text{on}(X, Y, T')))\]

- A block cannot be on itself \((\forall X, T') \neg\text{on}(X, X, T')\)
- The table cannot be on another block \((\forall Y, T') \neg\text{on}(\text{table}, Y, T')\)
- A block can only be on one block

\[(\forall X, Y_1, Y_2, T') (\text{on}(X, Y_1, T') \land \text{on}(X, Y_2, T') \rightarrow Y_1 = Y_2)\]

- A block except the table can support only one block

\[(\forall X_1, X_2, Y, T') (Y \neq \text{table} \land \text{on}(X_1, Y, T') \land \text{on}(X_2, Y, T') \rightarrow X_1 = X_2)\]
Action Axioms

- **Move X from Y to Z**

  \[(\forall X, Y, Z, T)(\text{on}(X, Y, T) \land \text{clear}(X, T) \land \text{clear}(Z, T) \land X \neq Y \land X \neq Z \land Y \neq Z \land X \neq \text{table} \land \text{move}(X, Y, Z, T) \rightarrow \text{on}(X, Z, T + 1) \land \text{clear}(Y, T + 1))\]

- **Actions are only executed if their preconditions hold**

  \[(\forall X, Y, Z, T)(\text{move}(X, Y, Z, T) \rightarrow \text{clear}(X, T) \land \text{clear}(Z, T) \land \text{on}(X, Y, T) \land X \neq Y \land X \neq Z \land Y \neq Z \land X \neq \text{table})\]
More Action Axioms

► Only one action occurs at a time

\[(\forall X_1, X_2, Y_1, Y_2, Z_1, Z_2, T)(\text{move}(X_1, Y_1, Z_1, T) \land \text{move}(X_2, Y_2, Z_2, T) \rightarrow X_1 = X_2 \land Y_1 = Y_2 \land Z_1 = Z_2)\]

► Some action occurs at every time

\[(\forall T)(\exists X, Y, Z) \text{move}(X, Y, Z, T)\]
Frame Axioms

► A clear block which is not covered as a result of a move action stays clear

\[(\forall X_1, X_2, Y, Z, T)(\text{clear}(X_2, T) \land \text{move}(X_1, Y, Z, T) \land X_2 \neq Y \land X_2 \neq Z \rightarrow \text{clear}(X_2, T + 1))\]

► A block stays on top of another one if it is not moved

\[(\forall X_1, X_2, Y_1, Y_2, Z, T)(\text{on}(X_2, Y_2, T) \land \text{move}(X_1, Y_1, Z, T) \land X_1 \neq X_2 \rightarrow \text{on}(X_2, Y_2, T + 1))\]
Let $\mathcal{A}$ be a set of action axioms, $\mathcal{F}$ be a set of frame axioms, $\mathcal{D}$ be a set of domain axioms, $\mathcal{E}$ be a set of equality axioms, $\mathcal{S}$ be an initial condition, $\mathcal{G}$ be a goal condition, then a planning problem is the question of whether

$$\mathcal{A} \cup \mathcal{F} \cup \mathcal{D} \cup \mathcal{E} \cup \{\mathcal{S}, \mathcal{G}\}$$

has a model
Example

- Let $a, b, \text{table}$ be all constants of type \textit{block}
- Let $[1, 3]$ be all constants of type \textit{time}
- Consider the planning problem
  
  \[ A \cup F \cup D \cup E \cup \{ \text{on}(a, b, 1) \land \text{on}(b, \text{table}, 1) \land \text{clear}(a, 1), \ \text{on}(b, a, 3) \} \]
- It has only one model (written as set instead of sequence)
  
  \[
  \{ \text{on}(a, b, 1), \ \text{on}(b, \text{table}, 1), \ \text{clear}(a, 1), \ \text{move}(a, b, \text{table}, 1), \\
  \text{on}(a, \text{table}, 2), \ \text{on}(b, \text{table}, 2), \ \text{clear}(a, 2), \ \text{clear}(b, 2), \ \text{move}(b, \text{table}, a, 2), \\
  \text{on}(a, \text{table}, 3), \ \text{on}(b, a, 3), \ \text{clear}(b, 3) \} \cup \{ \text{clear}(\text{table}, i) \mid 1 \leq i \leq 3 \} \cup E 
  \]
- We can extract the plan
  
  \[
  \text{move}(a, b, \text{table}, 1) \land \text{move}(b, \text{table}, a, 2) 
  \]
Remarks

Let $G$ be the specification of a planning problem

- Is $G$ correct?

- What is the meaning of “correct” in this context?

- If we consider (McCarthy, Hayes 1969), then at least one needs to
  - formally define the notion of a generated plan given a model of $G$ and
  - show that each generated plan is also a plan wrt the planning as deduction approach

- Is $G$ minimal?

- What are logical consequences of $G$?

- Reasoning is often easier in predicate logic
  - Reasoning with schemas as first-order formulas
  - But then we need to show that first-order satisfiability corresponds to propositional satisfiability
Solving Planning Problems

- Let $G$ be the specification of a planning problem
- $G$ can be solved using the following steps
  - Write $G$ in full
  - Transform $G$ into CNF
  - Bijectively replace ground atoms by propositional variables
  - Transform formulas into syntactic form required by a solver
  - Apply the solver
  - Read out the plan
- This will be demonstrated by means of our running example
Writing Specifications in Full

A block except the table cannot be clear and support a block at the same time

\[
(\forall X, Y, T') \ (Y \neq \text{table} \rightarrow \neg (\text{clear}(Y, T') \land \text{on}(X, Y, T')))
\]

is written in full as:

\[
\{ \ (a \neq \text{table} \rightarrow \neg (\text{clear}(a, 1) \land \text{on}(a, a, 1))), \\
(\ a \neq \text{table} \rightarrow \neg (\text{clear}(a, 2) \land \text{on}(a, a, 2))), \\
(\ a \neq \text{table} \rightarrow \neg (\text{clear}(a, 3) \land \text{on}(a, a, 3))), \\
(\ a \neq \text{table} \rightarrow \neg (\text{clear}(a, 1) \land \text{on}(b, a, 1))), \\
(\ a \neq \text{table} \rightarrow \neg (\text{clear}(a, 2) \land \text{on}(b, a, 2))), \\
(\ a \neq \text{table} \rightarrow \neg (\text{clear}(a, 3) \land \text{on}(b, a, 3))), \\
(\ a \neq \text{table} \rightarrow \neg (\text{clear}(a, 1) \land \text{on}(\text{table}, a, 1))), \\
(\ a \neq \text{table} \rightarrow \neg (\text{clear}(a, 2) \land \text{on}(\text{table}, a, 2))), \\
(\ a \neq \text{table} \rightarrow \neg (\text{clear}(a, 3) \land \text{on}(\text{table}, a, 3))), \\
(\ b \neq \text{table} \rightarrow \ldots \\
\ldots \\
\ldots \}
\]
Transformation in Conjunctive Normal Form

- A block except the table cannot be clear and support a block at the same time

\[(\forall X, Y, T') (Y \neq \text{table} \rightarrow \neg(\text{clear}(Y, T') \land \text{on}(X, Y, T'))]\]

- As CNF we obtain

\[
\langle
[a = \text{table}, \neg \text{clear}(a, 1), \neg \text{on}(a, a, 1)],
[a = \text{table}, \neg \text{clear}(a, 2), \neg \text{on}(a, a, 2)],
[a = \text{table}, \neg \text{clear}(a, 3), \neg \text{on}(a, a, 3)],
[a = \text{table}, \neg \text{clear}(a, 1), \neg \text{on}(b, a, 1)],
[a = \text{table}, \neg \text{clear}(a, 2), \neg \text{on}(b, a, 2)],
[a = \text{table}, \neg \text{clear}(a, 3), \neg \text{on}(b, a, 3)],
[a = \text{table}, \neg \text{clear}(a, 1), \neg \text{on}(\text{table}, a, 1)],
[a = \text{table}, \neg \text{clear}(a, 2), \neg \text{on}(\text{table}, a, 2)],
[a = \text{table}, \neg \text{clear}(a, 3), \neg \text{on}(\text{table}, a, 3)],
\ldots
\rangle
\]
Introduction of Propositional Variables

- A block except the table cannot be clear and support a block at the same time

\[(\forall X, Y, T') (Y \neq \text{table} \rightarrow \neg(\text{clear}(Y, T') \land \text{on}(X, Y, T')))]

- Replacing ground atoms by natural numbers we obtain

\[
\langle [3, \neg10, \neg19],
[3, \neg13, \neg28],
[3, \neg16, \neg37],
[3, \neg10, \neg22],
[3, \neg13, \neg31],
[3, \neg16, \neg40],
[3, \neg10, \neg25],
[3, \neg13, \neg34],
[3, \neg16, \neg43],
\ldots \rangle
\]
CNF-Form Required by the Solver

- A block except the table cannot be clear and support a block at the same time

\[(\forall X, Y, T') (Y \neq \text{table} \rightarrow \neg(\text{clear}(Y, T') \land \text{on}(X, Y, T')))\]

- The solver requires formulas to be in so-called .cnf-form

```
p cnf nv nc
3 -10 -19 0
3 -13 -28 0
3 -16 -37 0
3 -10 -22 0
3 -13 -31 0
3 -16 -40 0
3 -10 -25 0
3 -13 -34 0
3 -16 -43 0
```

where \text{nv} and \text{nc} are the number of variables and clauses, respectively
Application of a Solver

▶ Here we are applying the solver sat4j

▷ Check out the internet for sat4j

▷ In our example, \( nv = 99 \) and \( nc = 4299 \)

▷ It uses a different mapping from ground atoms to natural numbers

▷ It uses a different representation of interpretations
   atoms are listed iff they are mapped to \( \top \)

▷ It yields

\[ (1, 5, 9, 10, 11, 14, 15, 16, 17, 18, 22, 26, 27, 30, 34, 35, 56, 77) \]

▷ This translates into the model

\[
\begin{align*}
    & ( a = a, b = b, \text{table} = \text{table}, \\
    & \quad \text{clear}(a, 1), \text{clear}(a, 2), \text{clear}(b, 2), \text{clear}(b, 3), \\
    & \quad \text{clear}((\text{table}, 1), \text{clear}((\text{table}, 2), \text{clear}((\text{table}, 3), \\
    & \quad \text{on}(a, b, 1), \text{on}(a, \text{table}, 2), \text{on}(a, \text{table}, 3), \\
    & \quad \text{on}(b, a, 3), \text{on}(b, \text{table}, 1), \text{on}(b, \text{table}, 2), \\
    & \quad \text{move}(a, b, \text{table}, 1), \text{move}(b, \text{table}, a, 2) )
\end{align*}
\]
Reading out the Plan

▶ State at $t = 1$

\[ \langle \text{on}(a, b, 1), \text{on}(b, \text{table}, 1), \text{clear}(a, 1), \text{clear}(\text{table}, 1) \rangle \]

▶ Action at $t = 1$

\[ \text{move}(a, b, \text{table}, 1) \]

▶ State at $t = 2$

\[ \langle \text{clear}(a, 2), \text{clear}(b, 2), \text{clear}(\text{table}, 2), \text{on}(a, \text{table}, 2), \text{on}(b, \text{table}, 2) \rangle \]

▶ Action at $t = 2$

\[ \text{move}(b, \text{table}, a, 2) \]

▶ State at $t = 3$

\[ \langle \text{clear}(b, 3), \text{clear}(\text{table}, 3), \text{on}(a, \text{table}, 3), \text{on}(b, a, 3) \rangle \]
**Example: Periodic Event Scheduling Problems**

- Periodic events occur in traffic control systems, train scheduling systems and many other applications.
- The problem is to schedule periodic events with respect to some criteria.
- The problem is $NP$-complete.
- Real world problems are often very large.
  - Scheduling of trains in the railway network of Germany.
  - Only subnetworks can be dealt with currently.
- The previously best solvers were based on constraint programming techniques.
- We looked into a SAT-based approach.
Overview

- Periodic Event Networks
- Periodic Event Scheduling Problems
- Direct Encoding
- Order Encoding
- Experimental Evaluation
Intervals

Let \( I, U \in \mathbb{Z} \)

\[[I, U] = \{x \in \mathbb{Z} | I \leq x \leq U\}\) is the interval from \( I \) to \( U \)

\( I \) is called lower bound and \( U \) is called upper bound of the interval \([I, U]\)

Let \([I, U]\) be an interval and \( T \in \mathbb{N} \)

\([[I, U]]_T = \bigcup_{x \in \mathbb{Z}} [I + x \cdot T, U + x \cdot T]\) is called interval from \( I \) to \( U \) modulo \( T \)

\([[I, U]]_T \subseteq \mathbb{Z}\)

\([[2, 4]]_{10} = [[2, 4]] \cup [[12, 14]] \cup [−8, −6] \cup [[22, 24]] \cup [−18, −6] \cup \ldots\)

\([[I, U]]_0 = [I, U]\)
Let \((\mathcal{V}, \mathcal{E})\) be a graph, \(t \in \mathbb{N}\), and \(a : \mathcal{E} \rightarrow 2^{\mathbb{Z}}\) a mapping which assigns to each edge a finite set of intervals modulo \(t\).

- \(\mathcal{N} = (\mathcal{V}, \mathcal{E}, a, t)\) is called **periodic event network (PEN)**
- \(t\) is called **period**
- The elements of \(\mathcal{V}\) are called **(periodic) events**
- \(a(e)\) is called **set of constraints for the edge** \(e \in \mathcal{E}\)

Let \(\mathcal{N} = (\mathcal{V}, \mathcal{E}, a, t)\) be a PEN and \(\Pi : \mathcal{V} \rightarrow \mathbb{Z}\)

- \(\Pi\) is called **schedule for** \(\mathcal{N}\)

**SAT Problems 62**

Steffen Hölldobler and Norbert Manthey

INTERNATIONAL CENTER
FOR COMPUTATIONAL LOGIC
In PENs two types of constraints are usually distinguished: time consuming constraints and symmetry constraints.

Here, only time consuming constraints are considered.

Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, a, t)$ be a PEN, $(i, j) \in \mathcal{E}$, $[l, u]_t \in a(i, j)$, and $\Pi$ a schedule for $\mathcal{N}$.

- $[l, u]_t$ holds for $(i, j)$ under $\Pi$ iff $\Pi(j) - \Pi(i) \in [l, u]_t$.

A schedule $\Pi$ for a PEN $\mathcal{N}$ is said to be valid iff all constraints of $\mathcal{N}$ hold under $\Pi$. 
Example

Consider the following PEN $\mathcal{N}$

$\mathcal{N}$

Valid schedules for $\mathcal{N}$ are

$$\Pi_1 = \{p \mapsto 24, \ q \mapsto 27, \ r \mapsto 28, \ s \mapsto 30\}$$

$$\Pi_2 = \{p \mapsto 144, \ q \mapsto 147, \ r \mapsto 148, \ s \mapsto 150\}$$
Feasible Regions

Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, a, t)$ be a PEN and $(i, j) \in \mathcal{E}$

- Each $[l, u] \in a(i, j)$ constrains the possible values for $i$ and $j$ in a schedule
- Suppose $[3, 5] \subseteq a(i, j)$, then the blue regions are feasible, whereas the other regions are infeasible wrt the constraint $[3, 5]_{10}$
Equivalent Schedules

- Let $\Pi_1$ and $\Pi_2$ be schedules for the PEN $\mathcal{N} = (\mathcal{V}, \mathcal{E}, a, t)$
  - $\Pi_1$ and $\Pi_2$ are equivalent, in symbols $\Pi_1 \equiv \Pi_2$, iff for all $i \in \mathcal{V}$ we find $\Pi_1(i) \mod t = \Pi_2(i) \mod t$
  - Proposition $\equiv$ is an equivalence relation
  - Proposition If $\Pi_1 \equiv \Pi_2$ and $\Pi_1$ is valid, then $\Pi_2$ is also valid
  - Corollary If there exists a valid schedule $\Pi_1$ for $\mathcal{N}$, then there exists a valid schedule $\Pi_2 \equiv \Pi_1$ such that for all $i \in \mathcal{V}$ we find $\Pi_2(i) \in [0, t - 1]$
  - It suffices to search for schedules $\Pi$ with $\Pi(i) \in [0, t - 1]$ for all $i \in \mathcal{V}$
A periodic event scheduling problem (PESP) consists of a PEN $\mathcal{N}$ and is the question whether there exists a valid schedule for $\mathcal{N}$

- PESP is decidable
- PESP is $\mathcal{NP}$-complete
- If there exists a valid schedule, then the schedule shall be computed
Direct Encoding of Variables with Finite Domain

Let $x$ be a variable with finite domain $D$

- Variables are encoded with the help of propositional variables $p_{x,k}$ such that $p_{x,k}$ is mapped to $\top$ iff the value of $x$ is $k$
- The direct encoding of $x$ is

\[
\left( \bigvee_{k \in D} p_{x,k} \right) \land \left( \bigwedge_{k \in D} \bigwedge_{l \in D \setminus \{k\}} \neg (p_{x,k} \land p_{x,l}) \right)
\]

- The direct encoding of $x \in [2, 3]$ is

\[
(p_{x,2} \lor p_{x,3}) \land \neg (p_{x,2} \land p_{x,3}) \land \neg (p_{x,3} \land p_{x,2}) \\
\equiv (p_{x,2} \lor p_{x,3}) \land (\neg p_{x,2} \lor \neg p_{x,3})
\]
Direct Encoding of Values for Events

Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, a, t)$ be a PEN

- Each schedule $\Pi$ will assign a value from $[0, t - 1]$ to each $i \in \mathcal{V}$
- Hence, we obtain the following direct encoding for $\Pi(i)$

$$F_i = (\bigvee_{k \in [0, t-1]} p_{\Pi(i), k}) \land (\bigwedge_{k \in [0, t-1]} \bigwedge_{l \in [0, t-1] \setminus \{k\}} \neg(p_{\Pi(i), k} \land p_{\Pi(i), l}))$$

- Let

$$F_E = \bigwedge_{i \in \mathcal{V}} F_i$$
Direct Encoding of Time Consuming Constraints

- Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, a, t)$ be a PEN
  - Each constraint of $\mathcal{N}$ defines an infeasible region
  - Each infeasible region can be encoded as the negation of the disjunction of all points in the region
  - Let $\mathcal{F}_T$ be the conjunction of these encodings for all constraints in $\mathcal{N}$
- The direct encoding of a PEN $\mathcal{N}$ is
  $$\mathcal{F}_\mathcal{N} = \mathcal{F}_E \land \mathcal{F}_T$$
  - $\mathcal{F}_\mathcal{N}$ will be simplified and normalized before being submitted to a SAT-solver
We consider variables, whose domain is finite and ordered.

Here, we consider as domain intervals (modulo some $t \in \mathbb{N}$)

$x$ with domain $[1, 3]$

Variables are encoded with the help of propositional variables $q_{x,j}$ such that $q_{x,j}$ is mapped to $\top$ iff $x \leq j$

Let $x$ be a variable with domain $[l, u]$

The order encoding of $x$ is

$$\neg q_{x,l-1} \land q_{x,u} \land \bigwedge_{j \in [l, u]} (\neg q_{x,j-1} \lor q_{x,j})$$

The order encoding of $x$ with domain $[1, 3]$ is

$$\langle [\neg q_{x,0}, q_{x,3}], [\neg q_{x,0}, q_{x,1}], [\neg q_{x,1}, q_{x,2}], [\neg q_{x,2}, q_{x,3}] \rangle$$
Simplifying the Order Encoding

Recall

\[ \langle [-q_{x,0}, [q_{x,3}], [-q_{x,0}, q_{x,1}], [-q_{x,1}, q_{x,2}], [-q_{x,2}, q_{x,3}] \rangle = F \]

and observe that \([-q_{x,0}]\) and \([q_{x,3}]\) are unit clauses

Hence, any model for \(F\) must contain \([-q_{x,0}]\) and \([q_{x,3}]\), and

\[ F|_{(q_{x,3}, -q_{x,0})} = \langle [-q_{x,1}, q_{x,2}] \rangle. \]

Let \(x\) be a variable with domain \([l, u]\) and \(F_x\) its order encoding, then

\[ F_x|_{(q_u, -q_{x,l-1})} = \bigwedge_{j=[l+1,u-1]} (-q_{x,j-1} \lor q_{x,j}). \]

The latter is called simplified order encoding of \(x\)

The simplified order encoding of \(x\) with domain \([2, 3]\) or \([5, 5]\) is \(\langle \rangle\)
Order Encoding of Values for Events

Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, a, t)$ be a PEN

- Each schedule $\Pi$ will assign a value from $[0, t - 1]$ to each $i \in \mathcal{V}$
- Hence, we obtain the following order encoding for $\Pi(i)$

$$G_i = \neg q_{\Pi(i), t-1} \land q_{\Pi(i), t-1} \land \bigwedge_{j \in [1, t-1]} (\neg q_{\Pi(i), j-1} \lor q_{\Pi(i), j}).$$

Let

$$G_E = \bigwedge_{i \in \mathcal{V}} G_i.$$
Order Encoding of Time Consuming Constraints – Idea

Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, a, t)$ be a PEN, $(i, j) \in \mathcal{E}$, and $[3, 5]_{10} \in a(i, j)$

In the following figure, the red square is infeasible, i.e.,

$$\{ (\Pi(i), \Pi(j)) \mid \Pi(i) \in [4, 7], \Pi(j) \in [3, 6] \}$$

Idea Encode sufficiently many squares to cover the infeasible regions
Order Encoding an Infeasible Square

- Reconsider \{((\Pi(i), \Pi(j)) \mid \Pi(i) \in [4, 7], \Pi(j) \in [3, 6]\}

  - We obtain

    \[
    \neg(\Pi(i) \geq 4 \land \Pi(i) \leq 7 \land \Pi(j) \geq 3 \land \Pi(j) \leq 6)
    \equiv \neg(\neg\Pi(i) < 4 \land \Pi(i) \leq 7 \land \neg\Pi(j) < 3 \land \Pi(j) \leq 6)
    \equiv \neg(\neg\Pi(i) \leq 3 \land \Pi(i) \leq 7 \land \neg\Pi(j) \leq 2 \land \Pi(j) \leq 6)
    \equiv (\Pi(i) \leq 3 \lor \neg\Pi(i) \leq 7 \lor \Pi(j) \leq 2 \lor \neg\Pi(j) \leq 6)
    = [q_{\Pi(i),3}, \neg q_{\Pi(i),7}, q_{\Pi(j),2}, \neg q_{\Pi(j),6}]
    \]

  - The final formula is the encoding of the given infeasible square

- Suppose \([i, j]_t\) was the \(k\)th constraint of \(a(i, j)\) wrt some PEN \(\mathcal{N}\) (assuming some ordering)

  - Let \(G_{ijk}\) denote the conjunction of encodings of infeasible squares necessary to cover the infeasible regions wrt \([i, j]_t\)
Order Encoding of Time Consuming Constraints

Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, a, t)$ be a PEN

$$G_T = \bigwedge_{i \in \mathcal{V}} \bigwedge_{j \in \mathcal{V}} \bigwedge_{k \in a(i,j)} G_{ijk}$$

is the encoding of the time consuming constraints of $\mathcal{N}$

The order encoding of a PEN $\mathcal{N}$ is

$$G_\mathcal{N} = G_E \land G_T$$

$G_\mathcal{N}$ will be simplified and normalized before being submitted to a SAT-solver
Experimental Evaluation

- Cooperation with the Traffic Flow Science Group at the Faculty of Transportation and Traffic Science of TU Dresden
- Based on data from the Deutsche Bahn AG
- We compared
  - PESPsolve, a state-of-the-art constraint-based PESP-solver
  - Direct+RISS, the state-of-the-art SAT-solver RISS using direct encoding
  - Ordered+RISS, RISS using ordered encoding
- All solvers were given a timeout of 24h = 86400s
- The experiments were run on a Intel Core i7 with 8 GB RAM
### Number of Variables and Clauses

<table>
<thead>
<tr>
<th>instance</th>
<th></th>
<th></th>
<th>direct encoding $F_N$</th>
<th></th>
<th>order encoding $G_N$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$</td>
<td>\mathcal{V}</td>
<td>$</td>
<td>$#a$</td>
<td>$\text{var}(F_N)$</td>
</tr>
<tr>
<td>$\text{swg}_2$</td>
<td>60</td>
<td>1,145</td>
<td>7,200</td>
<td>2,037,732</td>
<td>7,140</td>
</tr>
<tr>
<td>$\text{fernsym}$</td>
<td>128</td>
<td>3,117</td>
<td>15,360</td>
<td>6,657,955</td>
<td>15,232</td>
</tr>
<tr>
<td>$\text{swg}_4$</td>
<td>170</td>
<td>7,107</td>
<td>20,400</td>
<td>6,193,570</td>
<td>20,230</td>
</tr>
<tr>
<td>$\text{swg}_3$</td>
<td>180</td>
<td>2,998</td>
<td>21,600</td>
<td>4,874,144</td>
<td>21,420</td>
</tr>
<tr>
<td>$\text{swg}_1$</td>
<td>221</td>
<td>7,443</td>
<td>26,520</td>
<td>7,601,906</td>
<td>26,299</td>
</tr>
<tr>
<td>$\text{seg}_2$</td>
<td>611</td>
<td>9,863</td>
<td>73,320</td>
<td>25,101,341</td>
<td>72,709</td>
</tr>
<tr>
<td>$\text{seg}_1$</td>
<td>1,483</td>
<td>10,351</td>
<td>177,960</td>
<td>34,323,942</td>
<td>176,477</td>
</tr>
</tbody>
</table>

**Notation**

- $\#a$ denotes the number of constraints given by $a$
- $\text{var}(X)$ denotes the number of variables occurring in $X$
- $|X|$ denotes the cardinality of the set $X$
## Results

<table>
<thead>
<tr>
<th>instance</th>
<th>PESPSOLVE/s</th>
<th>DIRECT+RISS/s</th>
<th>ORDERED+RISS/s</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>swg3</td>
<td>66</td>
<td>50</td>
<td>2</td>
<td>33</td>
</tr>
<tr>
<td>swg2</td>
<td>512</td>
<td>37</td>
<td>2</td>
<td>256</td>
</tr>
<tr>
<td>swg4</td>
<td>912</td>
<td>752</td>
<td>8</td>
<td>114</td>
</tr>
<tr>
<td>fernsym</td>
<td>2,035</td>
<td>294</td>
<td>7</td>
<td>290</td>
</tr>
<tr>
<td>swg1</td>
<td>TIMEOUT</td>
<td>18</td>
<td>7</td>
<td>&gt;12,342</td>
</tr>
<tr>
<td>seg1</td>
<td>TIMEOUT</td>
<td>16</td>
<td>10</td>
<td>&gt;8,640</td>
</tr>
<tr>
<td>seg2</td>
<td>TIMEOUT</td>
<td>TIMEOUT</td>
<td>11</td>
<td>&gt;7,854</td>
</tr>
</tbody>
</table>

**Conclusion**  The best PESP-solver is now SAT-based
Further Examples

► **Program Termination**
  In: *Proceedings SAT Conference*, LNCS 4501

► **Bioinformatics**
  Lynce, Marques-Silva 2008:
  Haplotype Inference with Boolean Satisfiability.
  In: *International Journal on Artificial Intelligence Tools* 17, 355-387

► **Bounded Model Checking**
  Clarke, Biere, Raimi, Zhu 2001:
  Bounded Model Checking using Satisfiability Solving.
  In: *Formal Methods in System Design* 19