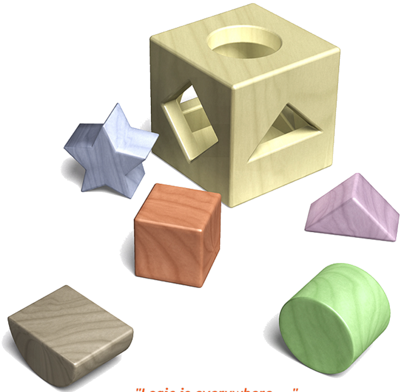
SAT Problems

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 $\{\neg/1,\ \wedge/2,\ \vee/2,\ \rightarrow/2,\ \leftrightarrow/2\}$ of connectives and
  - the special characters "(" and ")"
- $\cdot/n$ 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
  $\mathcal{F}, \mathcal{G}, \mathcal{H}$ 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** $\wedge^*/2$
  - **Disjunction** $\vee^*/2$
  - **Implication** $\rightarrow^*/2$
  - **Equivalence** $\leftrightarrow^*/2$

		$\neg^*$	$\wedge^*$	$\vee^*$	$\rightarrow^*$	$\leftrightarrow^*$
$\top$	$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\top$	$\bot$	$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\top$

Interpretations

- **Definition** An **interpretation** I consists of the set $\{\top, \bot\}$ and a mapping $\cdot^I : \mathcal{L}(\mathcal{R}) \rightarrow \{\top, \bot\}$ 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
  - I is a mapping
    - $\hat{I}$ does not contain a complementary pair of literals
  - 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:

$$\neg \succ \{\vee, \wedge\} \succ \rightarrow \succ \leftrightarrow$$

- 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

Propositional Satisfiability Problems

- **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 $F = 1$
  $\wedge\ (1 \vee 2)$
  $\wedge\ (1 \rightarrow 3)$
  $\wedge\ (1 \wedge 3 \rightarrow 4)$
  $\wedge\ (5 \vee 6)$
  $\wedge\ (5 \rightarrow 7)$
  $\wedge\ (\overline{5} \vee 8)$
  $\wedge\ (\overline{7} \vee \overline{8})$
- $(1, 2, 3, 4, \overline{5}, 6, \overline{7}, \overline{8})$ is a model for F
- Hence, F is satisfiable
- How can we find such a model?

Model Finding – First Ideas

- **Reconsider** $F = 1 \quad C_1$
  $\wedge (1 \vee 2) \quad C_2$
  $\wedge (1 \rightarrow 3) \quad C_3$
  $\wedge (1 \wedge 3 \rightarrow 4) \quad C_4$
  $\wedge (5 \vee 6) \quad C_5$
  $\wedge (5 \rightarrow 7) \quad C_6$
  $\wedge (\overline{5} \vee 8) \quad C_7$
  $\wedge (\overline{7} \vee \overline{8}) \quad C_8$

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 $J := (1, 3, 4, \dot{5})$
  - Because $5 \in J$ we find $J \models C_5$
  - Because $5 \in J$ and C_6 we set $J := (1.3.4, \dot{5}, 7)$, and thus $J \models C_6$
  - Because $5 \in J$ and C_7 we set $J := (1, 3, 4, \dot{5}, 7, 8)$ 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**

$$
\begin{array}{rll}
F = & 1 & C_1 \\
& \wedge\ (1 \vee 2) & C_2 \\
& \wedge\ (1 \rightarrow 3) & C_3 \\
& \wedge\ (1 \wedge 3 \rightarrow 4) & C_4 \\
& \wedge\ (5 \vee 6) & C_5 \\
& \wedge\ (5 \rightarrow 7) & C_6 \\
& \wedge\ (\overline{5} \vee 8) & C_7 \\
& \wedge\ (\overline{7} \vee \overline{8}). & C_8
\end{array}
$$

  - **Recall** $J := (1, 3, 4, \dot{5}, 7, 8)$ has led to a conflict
  - We backtrack and set $J := (1, 3, 4, \overline{5})$
  - Because $\overline{5} \in J$ and C_5 we set $J := (1, 3, 4, \overline{5}, 6)$ and thus $J \models C_5$
  - Because $\overline{5} \in J$ we find $J \models C_6$ and $J \models C_7$
  - In order to satisfy C_8 we **choose** $J := (1, 3, 4, \overline{5}, 6, \dot{\overline{7}})$ and thus $J \models C_8$
  - J is turned into a total interpretation by adding $2, \overline{8}$;
    the choice was arbitrary; we could have added $\overline{2}, \overline{8}$ or $2, 8$ or $\overline{2}, 8$

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 $\mathcal{S}(F)$ satisfying the following conditions:
  1. $F \in \mathcal{S}(F)$
  2. If $\neg G \in \mathcal{S}(F)$, then $G \in \mathcal{S}(F)$
  3. If $G_1 \circ G_2 \in \mathcal{S}(F)$, then $G_1,\ G_2 \in \mathcal{S}(F)$
- **Example**

$$\begin{aligned}&\mathcal{S}(\neg((p_1 \rightarrow p_2) \vee p_1)) \\ =\ &\{\neg((p_1 \rightarrow p_2) \vee p_1),\ ((p_1 \rightarrow p_2) \vee p_1),\ (p_1 \rightarrow p_2),\ p_1,\ p_2\}\end{aligned}$$

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	double negation
$\neg(F \wedge G)$	$\equiv$	$\neg F \vee \neg G$	
$\neg(F \vee G)$	$\equiv$	$\neg F \wedge \neg G$	de Morgan
$F \wedge (G \vee H)$	$\equiv$	$(F \wedge G) \vee (F \wedge H)$	
$F \vee (G \wedge H)$	$\equiv$	$(F \vee G) \wedge (F \vee H)$	distributivity
$F \leftrightarrow G$	$\equiv$	$(F \wedge G) \vee (\neg G \wedge \neg F)$	equivalence
$F \rightarrow G$	$\equiv$	$\neg F \vee G$	implication
$F \vee G$	$\equiv$	F, if F is valid	
$F \wedge G$	$\equiv$	G, if F is valid	tautology
$F \vee G$	$\equiv$	G, if F is unsatisfiable	
$F \wedge G$	$\equiv$	F, if F is unsatisfiable	unsatisfiability

Replacement

- $F\lceil G \mapsto H \rceil$ denotes the formula obtained from F by replacing an occurrence of $G \in \mathcal{S}(F)$ by H
  - Usually, the context determines which occurrence is meant
  - Sometimes the condition $G \in \mathcal{S}(F)$ is omitted
    In this case, if $G \notin \mathcal{S}(F)$, then $F\lceil G \mapsto H \rceil = F$
- **Replacement Theorem** If $G \equiv H$ then $F\lceil G \mapsto H \rceil \equiv F$

Generalized Disjunctions and Conjunctions

- **Generalized disjunction** $[F_1, \ldots, F_n] := F_1 \vee \ldots \vee F_n$
- **Generalized conjunction** $\langle F_1, \ldots, F_n \rangle := F_1 \wedge \ldots \wedge F_n$
- **Empty generalized disjunction** $[]$ with $[]^I = \bot$ for all I
- **Empty generalized conjunction** $\langle\rangle$ with $\langle\rangle^I = \top$ for all I
- **Note** $n \wedge \overline{n}$ is unsatisfiable, whereas $n \vee \overline{n}$ is valid, where $n \in \mathbb{N}^+$
- **Notation** We consider $\langle\rangle$ and $[]$ as abbreviations for $1 \vee \overline{1}$ and $1 \wedge \overline{1}$, 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 $C \vee L$ and $F \wedge C$, 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_i, $1 \leq i \leq n$, are said to be elements of $[L_1, \ldots, L_n]$ and
  - C_j, $1 \leq j \leq m$, are said to be elements of $\langle C_1, \ldots, C_m \rangle$

  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 lits

- Let lits be the following function from the set of clauses to the set of literals

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

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

$$\text{lits}(F) = \begin{cases} \emptyset & \text{if } F = \langle\rangle \\ \text{lits}(F') \cup \text{lits}(C). & \text{if } F = F', C \end{cases}$$

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

$$F \leftrightarrow G \quad \equiv \quad (F \wedge G) \vee (\neg F \wedge \neg G)$$

    - 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

$$F \rightarrow G \quad \equiv \quad \neg F \vee G$$

  - Hence, we assume that only the connectives $\neg$, $\wedge$ and $\vee$ 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

$$\frac{\neg\neg D}{D} \qquad \frac{(D_1 \wedge D_2)}{D_1 \mid D_2} \qquad \frac{\neg(D_1 \wedge D_2)}{\neg D_1, \neg D_2} \qquad \frac{(D_1 \vee D_2)}{D_1, D_2} \qquad \frac{\neg(D_1 \vee D_2)}{\neg D_1 \mid \neg D_2}$$

- 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 $F = p \wedge (p \rightarrow q) \rightarrow q$
- F is valid
- Eliminating implications yields

$$\neg(p \wedge (\neg p \vee q)) \vee q$$

- Applying the algorithm yields

$$\begin{array}{l}
\langle[\neg(p \wedge (\neg p \vee q)) \vee q]\rangle \\
\langle[\neg(p \wedge (\neg p \vee q)), q]\rangle \\
\langle[\neg p, \neg(\neg p \vee q), q]\rangle \\
\langle[\neg p, \neg\neg p \wedge \neg q, q]\rangle \\
\langle[\neg p, \neg\neg p, q], [\neg p, \neg q, q]\rangle \\
\langle[\neg p, p, q], [\neg p, \neg q, q]\rangle
\end{array}$$

- Both clauses in the final formula contain a complementary pair of literals

Remarks

- An application of a rule of the form $\frac{D}{D_1 \mid D_2}$ 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, $G \in \mathcal{S}(F)$ and $p \notin \mathcal{S}(F)$ a propositional variable
  - Replace F by $F\lceil G \mapsto p \rceil \wedge (p \leftrightarrow G)$
- Some observations
  - $F \not\equiv F\lceil G \mapsto p \rceil \wedge (p \leftrightarrow G)$
  - F is satisfiable iff $F\lceil G \mapsto p \rceil \wedge (p \leftrightarrow G)$ 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:

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

Then,

$$\begin{array}{lcl}
F|_{(1)} & = & \langle [3], [\overline{3},4], [5,6], [\overline{5},7], [\overline{5},8], [\overline{7},\overline{8}] \rangle \\
F|_{(1,3)} & = & \langle [4], [5,6], [\overline{5},7], [\overline{5},8], [\overline{7},\overline{8}] \rangle \\
F|_{(1,3,4)} & = & \langle [5,6], [\overline{5},7], [\overline{5},8], [\overline{7},\overline{8}] \rangle \\
F|_{(1,3,4,\overline{5})} & = & \langle [6], [\overline{7},\overline{8}] \rangle \\
F|_{(1,3,4,\overline{5},6)} & = & \langle [\overline{7},\overline{8}] \rangle \\
F|_{(1,3,4,\overline{5},6,\overline{7})} & = & \langle\ \rangle
\end{array}$$

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 $\overline{L}$;
The (propositional) resolvent of C_1 and C_2 with respect to L is the clause

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

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 6×10^{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

$$\bigwedge_{y=1}^{n^2} \bigwedge_{v=1}^{n^2} \bigwedge_{x=1}^{n^2-1} \bigwedge_{w=x+1}^{n^2} ((x, y, v) \to \neg(w, y, v))$$

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

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

- **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** McCarthy, Hayes: Some Philosophical Problems from the Standpoint of Artificial Intelligence. In: Machine Intelligence 4, Meltzer and Michie eds., Edinburgh University Press, 463-502: 1969
  - 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 move(a, b) in state s_1

$$\mathrm{on}(a, b, \mathrm{result}(\mathrm{move}(a, b), s_1))$$

  - 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

$$\neg \succ \{\vee, \wedge\} \succ \rightarrow \succ \leftrightarrow \succ \{\forall, \exists\}$$

  - 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**

$$\{a \neq b \mid a \text{ and } b \text{ are two different constants of type block}\}$$

Initial and Goal Conditions

- 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 time
  - $\mathrm{on}(a, b, 1) \wedge \mathrm{on}(b, \mathrm{table}, 1) \wedge \mathrm{clear}(a, 1)$
- Goal conditions are formulas in which only n appears as term of type time
  - With $n = 3$ we may consider $\mathrm{on}(b, a, 3)$

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

$$(\forall X, Y, T')\ (Y \neq \text{table} \to \neg(\text{clear}(Y, T') \land \text{on}(X, Y, T')))$$

- 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

$$(\forall X, Y_1, Y_2, T')\ (\text{on}(X, Y_1, T') \land \text{on}(X, Y_2, T') \to Y_1 = Y_2)$$

- A block except the table can support only one block

$$(\forall X_1, X_2, Y, T')\ (Y \neq \text{table} \land \text{on}(X_1, Y, T') \land \text{on}(X_2, Y, T') \to X_1 = X_2)$$

Action Axioms

- **Move X from Y to Z**

$$
\begin{aligned}
(\forall X, Y, Z, T)(\mathrm{on}(X, Y, T) \land \mathrm{clear}(X, T) \land \mathrm{clear}(Z, T) \\
\land X \neq Y \land X \neq Z \land Y \neq Z \land X \neq \mathrm{table} \\
\land \mathrm{move}(X, Y, Z, T) \\
\rightarrow \ \mathrm{on}(X, Z, T + 1) \land \mathrm{clear}(Y, T + 1))
\end{aligned}
$$

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

$$
\begin{aligned}
(\forall X, Y, Z, T)(\mathrm{move}(X, Y, Z, T) \\
\rightarrow \ \mathrm{clear}(X, T) \land \mathrm{clear}(Z, T) \land \mathrm{on}(X, Y, T) \\
\land X \neq Y \land X \neq Z \land Y \neq Z \land X \neq \mathrm{table})
\end{aligned}
$$

More Action Axioms

- Only one actions occurs at a time

$$(\forall X_1, X_2, Y_1, Y_2, Z_1, Z_2, T)(\mathrm{move}(X_1, Y_1, Z_1, T) \wedge \mathrm{move}(X_2, Y_2, Z_2, T) \\ \rightarrow X_1 = X_2 \wedge Y_1 = Y_2 \wedge Z_1 = Z_2)$$

- Some action occurs at every time

$$(\forall T)(\exists X, Y, Z)\ \mathrm{move}(X, Y, Z, T)$$

Frame Axioms

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

$$
\begin{aligned}
(\forall X_1, X_2, Y, Z, T)(\mathrm{clear}(X_2, T) \wedge \mathrm{move}(X_1, Y, Z, T) \\
\wedge X_2 \neq Y \wedge X_2 \neq Z \\
\rightarrow \ \mathrm{clear}(X_2, T+1))
\end{aligned}
$$

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

$$
\begin{aligned}
(\forall X_1, X_2, Y_1, Y_2, Z, T)(\mathrm{on}(X_2, Y_2, T) \wedge \mathrm{move}(X_1, Y_1, Z, T) \wedge X_1 \neq X_2 \\
\rightarrow \ \mathrm{on}(X_2, Y_2, T+1))
\end{aligned}
$$

Planning as Satisfiability Testing

- 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,
  S be an initial condition,
  G be a goal condition,

  then a planning problem is the question of whether

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

  has a model

Example

- Let a, b, table be all constants of type block
- Let $[1, 3]$ be all constants of type time
- Consider the planning problem

$$\mathcal{A} \cup \mathcal{F} \cup \mathcal{D} \cup \mathcal{E} \ \cup \{\text{on}(a, b, 1) \wedge \text{on}(b, \text{table}, 1) \wedge \text{clear}(a, 1),\ \text{on}(b, a, 3)\}$$

- It has only one model (written as set instead of sequence)

$$\begin{array}{l} \{\ \text{on}(a, b, 1),\ \text{on}(b, \text{table}, 1),\ \text{clear}(a, 1),\ \text{move}(a, b, \text{table}, 1), \\ \quad \text{on}(a, \text{table}, 2),\ \text{on}(b, \text{table}, 2), \text{clear}(a, 2),\ \text{clear}(b, 2),\ \text{move}(b, \text{table}, a, 2), \\ \quad \text{on}(a, \text{table}, 3),\ \text{on}(b, a, 3),\ \text{clear}(b, 3)\} \\ \cup \{\text{clear}(\text{table}, i) \mid 1 \leq i \leq 3\} \cup \mathcal{E} \end{array}$$

- We can extract the plan

$$\text{move}(a, b, \text{table}, 1) \wedge \text{move}(b, \text{table}, a, 2)$$

Remarks

Let $\mathcal{G}$ be the specification of a planning problem

- Is $\mathcal{G}$ 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 $\mathcal{G}$ and
  - show that each generated plan is also a plan wrt the planning as deduction approach
- Is $\mathcal{G}$ minimal?
- What are logical consequences of $\mathcal{G}$?
- 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 $\mathcal{G}$ be the specification of a planning problem
- $\mathcal{G}$ can be solved using the following steps
  - Write $\mathcal{G}$ in full
  - Transform $\mathcal{G}$ into CNF
  - Bijectively replace ground atoms by propositional variables
  - Transform formulas into syntatic 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**

$$(\forall X, Y, T')\,(Y \neq \text{table} \rightarrow \neg(\text{clear}(Y, T') \wedge \text{on}(X, Y, T')))$$

- **is written in full as:**

$$\begin{array}{ll}
\{ & (a \neq \text{table} \rightarrow \neg(\text{clear}(a, 1) \wedge \text{on}(a, a, 1))), \\
 & (a \neq \text{table} \rightarrow \neg(\text{clear}(a, 2) \wedge \text{on}(a, a, 2))), \\
 & (a \neq \text{table} \rightarrow \neg(\text{clear}(a, 3) \wedge \text{on}(a, a, 3))), \\
 & (a \neq \text{table} \rightarrow \neg(\text{clear}(a, 1) \wedge \text{on}(b, a, 1))), \\
 & (a \neq \text{table} \rightarrow \neg(\text{clear}(a, 2) \wedge \text{on}(b, a, 2))), \\
 & (a \neq \text{table} \rightarrow \neg(\text{clear}(a, 3) \wedge \text{on}(b, a, 3))), \\
 & (a \neq \text{table} \rightarrow \neg(\text{clear}(a, 1) \wedge \text{on}(\text{table}, a, 1))), \\
 & (a \neq \text{table} \rightarrow \neg(\text{clear}(a, 2) \wedge \text{on}(\text{table}, a, 2))), \\
 & (a \neq \text{table} \rightarrow \neg(\text{clear}(a, 3) \wedge \text{on}(\text{table}, a, 3))), \\
 & (b \neq \text{table} \rightarrow \ldots \\
 & \vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \}
\end{array}$$

Transformation in Conjunctive Normal Form

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

$$(\forall X, Y, T')\,(Y \neq \text{table} \rightarrow \neg(\text{clear}(Y, T') \wedge \text{on}(X, Y, T')))$$

- **As CNF we obtain**

$$\begin{array}{ll}\langle & [a = \text{table}, \neg\text{clear}(a, 1), \neg\text{on}(a, a, 1)],\\ & [a = \text{table}, \neg\text{clear}(a, 2), \neg\text{on}(a, a, 2)],\\ & [a = \text{table}, \neg\text{clear}(a, 3), \neg\text{on}(a, a, 3)],\\ & [a = \text{table}, \neg\text{clear}(a, 1), \neg\text{on}(b, a, 1)],\\ & [a = \text{table}, \neg\text{clear}(a, 2), \neg\text{on}(b, a, 2)],\\ & [a = \text{table}, \neg\text{clear}(a, 3), \neg\text{on}(b, a, 3)],\\ & [a = \text{table}, \neg\text{clear}(a, 1), \neg\text{on}(\text{table}, a, 1)],\\ & [a = \text{table}, \neg\text{clear}(a, 2), \neg\text{on}(\text{table}, a, 2)],\\ & [a = \text{table}, \neg\text{clear}(a, 3), \neg\text{on}(\text{table}, a, 3)],\\ & \vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \rangle\end{array}$$

Introduction of Propositional Variables

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

$$(\forall X, Y, T')\,(Y \neq \text{table} \rightarrow \neg(\text{clear}(Y, T') \wedge \text{on}(X, Y, T')))$$

- Replacing ground atoms by natural numbers we obtain

$$\begin{array}{ll} \langle & [3, \neg 10, \neg 19], \\ & [3, \neg 13, \neg 28], \\ & [3, \neg 16, \neg 37], \\ & [3, \neg 10, \neg 22], \\ & [3, \neg 13, \neg 31], \\ & [3, \neg 16, \neg 40], \\ & [3, \neg 10, \neg 25], \\ & [3, \neg 13, \neg 34], \\ & [3, \neg 16, \neg 43], \\ & \vdots \qquad\qquad\qquad \rangle \end{array}$$

CNF-Form Required by the Solver

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

$$(\forall X, Y, T')\,(Y \neq \text{table} \rightarrow \neg(\text{clear}(Y, T') \wedge \text{on}(X, Y, T')))$$

- **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 nv and 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
$$\begin{array}{ll}(& a = a, b = b, \text{table} = \text{table}, \\ & \text{clear}(a, 1), \text{clear}(a, 2), \text{clear}(b, 2), \text{clear}(b, 3), \\ & \text{clear}(\text{table}, 1), \text{clear}(\text{table}, 2), \text{clear}(\text{table}, 3), \\ & \text{on}(a, b, 1), \text{on}(a, \text{table}, 2), \text{on}(a, \text{table}, 3), \\ & \text{on}(b, a, 3), \text{on}(b, \text{table}, 1), \text{on}(b, \text{table}, 2), \\ & \text{move}(a, b, \text{table}, 1), \text{move}(b, \text{table}, a, 2) \qquad)\end{array}$$

Reading out the Plan

- **State at $t = 1$**

$$\langle \mathrm{on}(a, b, 1), \mathrm{on}(b, \mathrm{table}, 1), \mathrm{clear}(a, 1), \mathrm{clear}(\mathrm{table}, 1)\rangle$$

- **Action at $t = 1$**

$$\mathrm{move}(a, b, \mathrm{table}, 1)$$

- **State at $t = 2$**

$$\langle \mathrm{clear}(a, 2), \mathrm{clear}(b, 2), \mathrm{clear}(\mathrm{table}, 2), \mathrm{on}(a, \mathrm{table}, 2), \mathrm{on}(b, \mathrm{table}, 2)\rangle$$

- **Action at $t = 2$**

$$\mathrm{move}(b, \mathrm{table}, a, 2)$$

- **State at $t = 3$**

$$\langle \mathrm{clear}(b, 3), \mathrm{clear}(\mathrm{table}, 3), \mathrm{on}(a, \mathrm{table}, 3), \mathrm{on}(b, a, 3)\rangle$$

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 $\mathcal{NP}$-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**
  - **Großmann, H., Manthey, Nachtigall, Opitz, Steinke: Solving Periodic Event Scheduling Problems with SAT. In: Advanced Research in Applied Artificial Intelligence, LNCS 7345, 166-175: 2012**

Overview

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

Intervals

- Let $l, u \in \mathbb{Z}$
  - $[l, u] = \{x \in \mathbb{Z} \mid l \leq x \leq u\}$ is the interval from l to u
  - l is called lower bound and u is called upper bound of the interval $[l, u]$
- Let $[l, u]$ be an interval and $t \in \mathbb{N}$
  - $[l, u]_t = \bigcup_{x \in \mathbb{Z}} [l + x \cdot t, u + x \cdot t]$ is called interval from l to u modulo t
  - $[l, u]_t \subseteq \mathbb{Z}$
  - $[2, 4]_{10} = [2, 4] \cup [12, 14] \cup [-8, -6] \cup [22, 24] \cup [-18, -6] \cup \ldots$
  - $[l, u]_0 = [l, u]$

Periodic Event Networks and Schedules

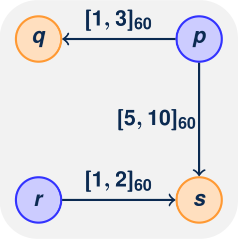
- Let $(\mathcal{V}, \mathcal{E})$ be a graph, $t \in \mathbb{N}$, and $a : \mathcal{E} \rightarrow 2^{2^{\mathbb{Z}}}$ a mapping which assigns to each edge a finite set of intervals modulo t
  - $\mathcal{N} = (\mathcal{V}, \mathcal{E}, a, t)$ is called periodic event network (PEN)
  - t is called period
  - 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 $\Pi : \mathcal{V} \rightarrow \mathbb{Z}$
  - Π is called schedule for $\mathcal{N}$

Constraints

- 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 Π a schedule for $\mathcal{N}$
  - $[l, u]_t$ holds for (i, j) under Π iff $\Pi(j) - \Pi(i) \in [l, u]_t$
- A schedule Π for a PEN $\mathcal{N}$ is said to be valid iff all constraints of $\mathcal{N}$ hold under Π

Example

- Consider the following PEN $\mathcal{N}$

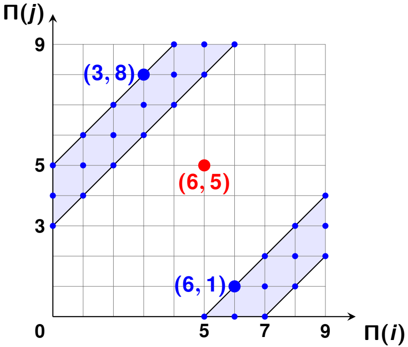


- Valid schedules for $\mathcal{N}$ are

$$\begin{array}{lcl} \Pi_1 & = & \{p \mapsto 24,\ q \mapsto 27,\ r \mapsto 28,\ s \mapsto 30\} \\ \Pi_2 & = & \{p \mapsto 144,\ q \mapsto 147,\ r \mapsto 148,\ s \mapsto 150\} \end{array}$$

Feasible Regions

- Let $\mathcal{N} = (\mathcal{V}, \mathcal{E}, a, t)$ be a PEN and $(i, j) \in \mathcal{E}$
  - Each $[l, u]_t \in a(i, j)$ constrains the possible values for i and j in a schedule
  - Suppose $[3, 5]_{10} \in a(i, j)$, then the blue regions are feasible, whereas the other regions are infeasible wrt the constraint $[3, 5]_{10}$



Equivalent Schedules

- Let Π_1 and Π_2 be schedules for the PEN $\mathcal{N} = (\mathcal{V}, \mathcal{E}, a, t)$
  - Π_1 and Π_2 are **equivalent**, in symbols $\Pi_1 \equiv \Pi_2$,
    iff for all $i \in \mathcal{V}$ we find $\Pi_1(i) \bmod t = \Pi_2(i) \bmod t$
  - **Proposition** $\equiv$ is an equivalence relation
  - **Proposition** If $\Pi_1 \equiv \Pi_2$ and Π_1 is valid, then Π_2 is also valid
  - **Corollary** If there exists a valid schedule Π_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 Π with $\Pi(i) \in [0, t-1]$ for all $i \in \mathcal{V}$

Periodic Event Scheduling Problems

- 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
  - Until 2011 the best PESP-solvers were based on constraint propagation techniques (Opitz: Automatische Erzeugung und Optimierung von Taktfahrplänen in Schienenverkehrsnetzen. PhD thesis, TU Dresden: 2009)

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

$$(\bigvee_{k \in D} p_{x,k}) \wedge (\bigwedge_{k \in D} \bigwedge_{l \in D \setminus \{k\}} \neg(p_{x,k} \wedge p_{x,l}))$$

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

$$\begin{array}{ll} & (p_{x,2} \vee p_{x,3}) \wedge \neg(p_{x,2} \wedge p_{x,3}) \wedge \neg(p_{x,3} \wedge p_{x,2}) \\ \equiv & (p_{x,2} \vee p_{x,3}) \wedge (\neg p_{x,2} \vee \neg p_{x,3}) \end{array}$$

Direct Encoding of Values for Events

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

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

  - Let

$$\mathcal{F}_E = \bigwedge_{i \in \mathcal{V}} F_i$$

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

$$\mathcal{F}_\mathcal{N} = \mathcal{F}_E \wedge \mathcal{F}_T$$

  - $\mathcal{F}_\mathcal{N}$ will be simplified and normalized before being submitted to a SAT-solver

Encoding Variables with Finite Ordered Domain

- 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

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

  - 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**

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

and observe that $[\neg q_{x,0}]$ and $[q_{x,3}]$ are unit clauses

- **Hence, any model for F must contain $\neg q_{x,0}$ and $q_{x,3}$, and**

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

- **Let x be a variable with domain $[l, u]$ and F_x its order encoding, then**

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

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

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

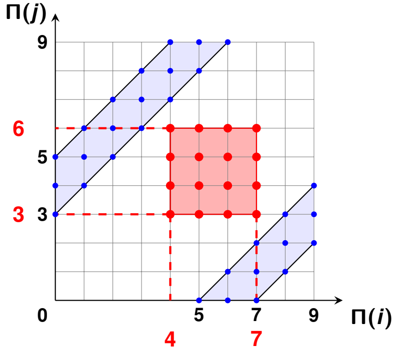
  - Let

$$\mathcal{G}_E = \bigwedge_{i \in \mathcal{V}} G_i$$

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**

$$
\begin{array}{ll}
 & \neg(\Pi(i) \geq 4 \wedge \Pi(i) \leq 7 \wedge \Pi(j) \geq 3 \wedge \Pi(j) \leq 6) \\
\equiv & \neg(\neg\Pi(i) < 4 \wedge \Pi(i) \leq 7 \wedge \neg\Pi(j) < 3 \wedge \Pi(j) \leq 6) \\
\equiv & \neg(\neg\Pi(i) \leq 3 \wedge \Pi(i) \leq 7 \wedge \neg\Pi(j) \leq 2 \wedge \Pi(j) \leq 6) \\
\equiv & (\Pi(i) \leq 3 \vee \neg\Pi(i) \leq 7 \vee \Pi(j) \leq 2 \vee \neg\Pi(j) \leq 6) \\
= & [q_{\Pi(i),3}, \neg q_{\Pi(i),7}, q_{\Pi(j),2}, \neg q_{\Pi(j),6}]
\end{array}
$$

  The final formula is the encoding of the given infeasible square

- **Suppose $[i, j]_t$ was the kth 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

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

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

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

$$\mathcal{G}_{\mathcal{N}} = \mathcal{G}_E \wedge \mathcal{G}_T$$

  - $\mathcal{G}_{\mathcal{N}}$ 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 constaint-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

	$\mathcal{N} = (\mathcal{V}, \mathcal{E}, a, t)$		direct encoding $\mathcal{F}_{\mathcal{N}}$		order encoding $\mathcal{G}_{\mathcal{N}}$	
instance	$\lvert\mathcal{V}\rvert$	$\#a$	$\lvert\mathrm{var}(\mathcal{F}_{\mathcal{N}})\rvert$	$\lvert\mathcal{F}_{\mathcal{N}}\rvert$	$\lvert\mathrm{var}(\mathcal{G}_{\mathcal{N}})\rvert$	$\lvert\mathcal{G}_{\mathcal{N}}\rvert$
swg_2	60	1,145	7,200	2,037,732	7,140	83,740
fernsym	128	3,117	15,360	6,657,955	15,232	353,276
swg_4	170	7,107	20,400	6,193,570	20,230	399,191
swg_3	180	2,998	21,600	4,874,144	21,420	214,011
swg_1	221	7,443	26,520	7,601,906	26,299	462,217
seg_2	611	9,863	73,320	25,101,341	72,709	1,115,210
seg_1	1,483	10,351	177,960	34,323,942	176,477	1,348,045

- **Notation**
  - $\#a$ denotes the number of constraints given by a
  - $\mathrm{var}(X)$ denotes the number of variables occurring in X
  - $\lvert X\rvert$ denotes the cardinality of the set X

Results

instance	PESPSOLVE/s	DIRECT+RISS/s	ORDERED+RISS/s	speedup
swg_3	66	50	2	33
swg_2	512	37	2	256
swg_4	912	752	8	114
fernsym	2,035	294	7	290
swg_1	TIMEOUT	18	7	$>$12,342
seg_1	TIMEOUT	16	10	$>$8,640
seg_2	TIMEOUT	TIMEOUT	11	$>$7,854

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

Further Examples

- **Program Termination**
  Fuhs, Giesl, Middeldorp, Schneider-Kamp, Thiemann, Zankl 2007:
  SAT Solving for Termination Analysis with Polynomial Interpretations.
  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