Answer Set Solving in Practice

Martin Gebser and Torsten Schaub
University of Potsdam
torsten@cs.uni-potsdam.de

Rough Roadmap

1. Introduction
2. Language
3. Modeling
4. Grounding
5. Foundations
6. Solving
7. Systems
8. Applications

Potassco

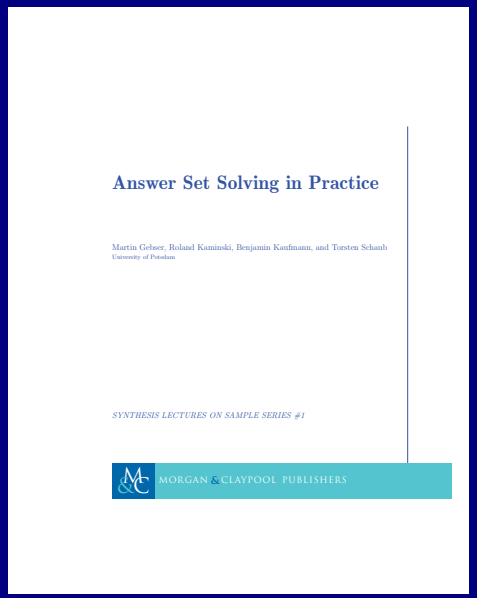
Resources

- Course material
  - http://www.cs.uni-potsdam.de/wv/lehre
  - http://moodle.cs.uni-potsdam.de
  - http://potassco.sourceforge.net/teaching.html
- Systems
  - clasp http://potassco.sourceforge.net
  - dlv http://www.dlvsystem.com
  - smodels http://www.tcs.hut.fi/Software/smodels
  - gringo http://potassco.sourceforge.net
  - lparse http://www.tcs.hut.fi/Software/smodels
  - clingo http://potassco.sourceforge.net
  - iclingo http://potassco.sourceforge.net
  - oclingo http://potassco.sourceforge.net
  - asparagus http://asparagus.cs.uni-potsdam.de

Potassco

The Potassco Book

1. Motivation
2. Introduction
3. Basic modeling
4. Grounding
5. Characterizations
6. Solving
7. Systems
8. Advanced modeling
9. Conclusions



Resources

- `http://potassco.sourceforge.net/book.html`
- `http://potassco.sourceforge.net/teaching.html`

Potassco

Literature

Books	[4], [29], [53]
Surveys	[50], [2], [39], [21], [11]
Articles	[41], [42], [6], [61], [54], [49], [40], etc.

Potassco

Constraint Answer Set Programming: Overview

1. Motivation
2. Preliminaries
3. Modeling Language
4. Algorithms

Potassco

Outline

Potassco

Motivation

- Observation Certain applications are more naturally modeled by mixing Boolean with non-Boolean constructs, eg., accounting for
  - resources,
  - fine timings, or
  - functions over finite domains
- Approach Extend ASP with CP capacities
  - propositional semantics
  - CDCL-style solving capacities
    - following the approach of SMT solving
  - off-the-shelf CP solvers

Potassco

Motivation

- Observation Certain applications are more naturally modeled by mixing Boolean with non-Boolean constructs, eg., accounting for
  - resources,
  - fine timings, or
  - functions over finite domains
- Approach Extend ASP with CP capacities
  - propositional semantics
  - CDCL-style solving capacities
    - following the approach of SMT solving
  - off-the-shelf CP solvers

Potassco

Outline

Potassco

Constraint Satisfaction Problem

- A constraint satisfaction problem (CSP) consists of
  - a set V of variables,
  - a set D of domains, and
  - a set C of constraints

  such that
  - each variable $v \in V$ has an associated domain $dom(v) \in D$;
  - a constraint c is a pair (S, R) consisting of a k-ary relation R on a vector $S \subseteq V^k$ of variables, called the scope of R

- Note For $S = (v_1, \ldots, v_k)$, we have $R \subseteq dom(v_1) \times \cdots \times dom(v_k)$

Potassco

Constraint Satisfaction Problem

- A constraint satisfaction problem (CSP) consists of
  - a set V of variables,
  - a set D of domains, and
  - a set C of constraints

  such that
  - each variable $v \in V$ has an associated domain $dom(v) \in D$;
  - a constraint c is a pair (S, R) consisting of a k-ary relation R on a vector $S \subseteq V^k$ of variables, called the scope of R

- Note For $S = (v_1, \ldots, v_k)$, we have $R \subseteq dom(v_1) \times \cdots \times dom(v_k)$

Potassco

Constraint Satisfaction Problem

- A constraint satisfaction problem (CSP) consists of
  - a set V of variables,
  - a set D of domains, and
  - a set C of constraints

  such that
  - each variable $v \in V$ has an associated domain $dom(v) \in D$;
  - a constraint c is a pair (S, R) consisting of a k-ary relation R on a vector $S \subseteq V^k$ of variables, called the scope of R
- Note For $S = (v_1, \ldots, v_k)$, we have $R \subseteq dom(v_1) \times \cdots \times dom(v_k)$

Potassco

Example

$$\begin{array}{cccccc} & s & e & n & d \\ + & m & o & r & e \\ \hline m & o & n & e & y \end{array}$$

Each letter corresponds exactly to one digit and all variables have to be pairwisely distinct

$$V = \{s, e, n, d, m, o, r, y\}$$

$$D = \{dom(v) = \{0, \ldots, 9\} \mid v \in V\}$$

$$\begin{aligned} C = \{ &(\vec{V}, \quad allDistinct(V)), \\ &(\vec{V}, \quad s \times 1000 + e \times 100 + n \times 10 + d + \\ &\qquad m \times 1000 + o \times 100 + r \times 10 + e == \\ &\qquad m \times 10000 + o \times 1000 + n \times 100 + e \times 10 + y), \\ &((m), m == 1) \} \end{aligned}$$

Potassco

Example

$$\begin{array}{cccccc} & s & e & n & d \\ + & m & o & r & e \\ \hline m & o & n & e & y \end{array}$$

Each letter corresponds exactly to one digit and all variables have to be pairwisely distinct

$$V = \{s, e, n, d, m, o, r, y\}$$

$$D = \{dom(v) = \{0, \ldots, 9\} \mid v \in V\}$$

$$\begin{aligned} C = \{ &(\vec{V}, \quad allDistinct(V)), \\ &(\vec{V}, \quad s \times 1000 + e \times 100 + n \times 10 + d + \\ &\qquad\quad m \times 1000 + o \times 100 + r \times 10 + e == \\ &\qquad\quad m \times 10000 + o \times 1000 + n \times 100 + e \times 10 + y), \\ &((m), m == 1) \} \end{aligned}$$

Potassco

Example

	s	e	n	d
+	m	o	r	e
m	o	n	e	y

Each letter corresponds exactly to one digit and all variables have to be pairwisely distinct

	9	5	6	7
+	1	0	8	5
1	0	6	5	2

The example has exactly one solution

$$\{ s \mapsto 9, e \mapsto 5, n \mapsto 6, d \mapsto 7, m \mapsto 1, o \mapsto 0, r \mapsto 8, y \mapsto 2 \}$$

Potassco

Constraint satisfaction problem

- Notation We use $S(c) = S$ and $R(c) = R$ to access the scope and the relation of a constraint $c = (S, R)$
- For an assignment $A : V \to \bigcup_{v \in V} dom(v)$ and a constraint (S, R) with scope $S = (v_1, \ldots, v_k)$, define

$$sat_C(A) = \{c \in C \mid A(S(c)) \in R(c)\}$$

where $A(S) = (A(v_1), \ldots, A(v_k))$

Potassco

Constraint satisfaction problem

- Notation We use $S(c) = S$ and $R(c) = R$ to access the scope and the relation of a constraint $c = (S, R)$
- For an assignment $A : V \to \bigcup_{v \in V} dom(v)$ and a constraint (S, R) with scope $S = (v_1, \ldots, v_k)$, define

$$sat_C(A) = \{c \in C \mid A(S(c)) \in R(c)\}$$

where $A(S) = (A(v_1), \ldots, A(v_k))$

Potassco

Constraint Answer Set Programming

- A constraint logic program P is a logic program over an extended alphabet $\mathcal{A} \cup \mathcal{C}$ where
  - $\mathcal{A}$ is a set of regular atoms and
  - $\mathcal{C}$ is a set of constraint atoms,

  such that $\mathit{head}(r) \in \mathcal{A}$ for each $r \in P$
- Given a set of literals B and some set $\mathcal{B}$ of atoms, we define $B|_{\mathcal{B}} = (B^+ \cap \mathcal{B}) \cup \{\sim a \mid a \in B^- \cap \mathcal{B}\}$

Potassco

Constraint Answer Set Programming

- A constraint logic program P is a logic program over an extended alphabet $\mathcal{A} \cup \mathcal{C}$ where
  - $\mathcal{A}$ is a set of regular atoms and
  - $\mathcal{C}$ is a set of constraint atoms,

  such that $\mathit{head}(r) \in \mathcal{A}$ for each $r \in P$
- Given a set of literals B and some set $\mathcal{B}$ of atoms, we define $B|_{\mathcal{B}} = (B^+ \cap \mathcal{B}) \cup \{\sim a \mid a \in B^- \cap \mathcal{B}\}$

Potassco

Constraint Answer Set Programming

- We identify constraint atoms with constraints via a function

$$\gamma : \mathcal{C} \to C$$

- Furthermore, $\gamma(Y) = \{\gamma(c) \mid c \in Y\}$ for any $Y \subseteq \mathcal{C}$
- Note Unlike regular atoms $\mathcal{A}$, constraint atoms $\mathcal{C}$ are not subject to the unique names assumption, eg.

$$\gamma(x < y) \;=\; \gamma(((-y - 1) \leq -(x + 1)) \wedge (x \neq y))$$

- A constraint logic program P is associated with a CSP as follows
  - $C[P] = \gamma(atom(P) \cap \mathcal{C})$,
  - $V[P]$ is obtained from the constraint scopes in $C[P]$,
  - $D[P]$ is provided by a declaration

Potassco

Constraint Answer Set Programming

- We identify constraint atoms with constraints via a function

$$\gamma : \mathcal{C} \to C$$

- Furthermore, $\gamma(Y) = \{\gamma(c) \mid c \in Y\}$ for any $Y \subseteq \mathcal{C}$
- Note Unlike regular atoms $\mathcal{A}$, constraint atoms $\mathcal{C}$ are not subject to the unique names assumption, eg.

$$\gamma(x < y) \;=\; \gamma(((-y - 1) \leq -(x + 1)) \wedge (x \neq y))$$

- A constraint logic program P is associated with a CSP as follows
  - $C[P] = \gamma(atom(P) \cap \mathcal{C})$,
  - $V[P]$ is obtained from the constraint scopes in $C[P]$,
  - $D[P]$ is provided by a declaration

Potassco

Constraint Answer Set Programming

- We identify constraint atoms with constraints via a function

$$\gamma : \mathcal{C} \to C$$

- Furthermore, $\gamma(Y) = \{\gamma(c) \mid c \in Y\}$ for any $Y \subseteq \mathcal{C}$
- Note Unlike regular atoms $\mathcal{A}$, constraint atoms $\mathcal{C}$ are not subject to the unique names assumption, eg.

$$\gamma(x < y) \;=\; \gamma(((-y - 1) \leq -(x + 1)) \wedge (x \neq y))$$

- A constraint logic program P is associated with a CSP as follows
  - $C[P] = \gamma(atom(P) \cap \mathcal{C})$,
  - $V[P]$ is obtained from the constraint scopes in $C[P]$,
  - $D[P]$ is provided by a declaration

Potassco

Constraint Answer Set Programming

- Let P be a constraint logic program over $\mathcal{A} \cup \mathcal{C}$ and let $A : V[P] \to D[P]$ be an assignment,

  define the constraint reduct of as P wrt A as follows

$$P^A = \{\ head(r) \leftarrow body(r)|_{\mathcal{A}} \mid r \in P, \gamma(body(r)|_{\mathcal{C}}{}^{+}) \subseteq sat_{C[P]}(A), \gamma(body(r)|_{\mathcal{C}}{}^{-}) \cap sat_{C[P]}(A) = \emptyset\ \}$$

- A set $X \subseteq \mathcal{A}$ of (regular) atoms is a constraint answer set of P wrt A, if X is an answer set of P^A.
  - Note: That is, if X is the $\subseteq$-smallest model of $(P^A)^X$

Potassco

Constraint Answer Set Programming

- Let P be a constraint logic program over $\mathcal{A} \cup \mathcal{C}$ and let $A : V[P] \to D[P]$ be an assignment,

  define the constraint reduct of as P wrt A as follows

$$P^A = \{\, head(r) \leftarrow body(r)|_{\mathcal{A}} \mid r \in P, \gamma(body(r)|_{\mathcal{C}}{}^{+}) \subseteq sat_{C[P]}(A), \gamma(body(r)|_{\mathcal{C}}{}^{-}) \cap sat_{C[P]}(A) = \emptyset \,\}$$

- A set $X \subseteq \mathcal{A}$ of (regular) atoms is a constraint answer set of P wrt A, if X is an answer set of P^A.
- Note That is, if X is the $\subseteq$-smallest model of $(P^A)^X$

Potassco

Constraint Answer Set Programming

- Let P be a constraint logic program over $\mathcal{A} \cup \mathcal{C}$ and let $A : V[P] \to D[P]$ be an assignment,

  define the constraint reduct of as P wrt A as follows

$$P^A = \{\, head(r) \leftarrow body(r)|_{\mathcal{A}} \mid r \in P, \gamma(body(r)|_{\mathcal{C}}{}^{+}) \subseteq sat_{C[P]}(A), \gamma(body(r)|_{\mathcal{C}}{}^{-}) \cap sat_{C[P]}(A) = \emptyset \,\}$$

- A set $X \subseteq \mathcal{A}$ of (regular) atoms is a constraint answer set of P wrt A, if X is an answer set of P^A.
- Note That is, if X is the $\subseteq$-smallest model of $(P^A)^X$

Potassco

Outline

Potassco

Modeling Language

Note

Although our semantics is propositional, the atoms in $\mathcal{A}$ and $\mathcal{C}$ are constructible from a multi-sorted, first-order signature given by:

- a set $\mathcal{P}_\mathcal{A} \cup \mathcal{P}_\mathcal{C}$ of predicate symbols such that $\mathcal{P}_\mathcal{A} \cap \mathcal{P}_\mathcal{C} = \emptyset$,
- a set $\mathcal{F}_\mathcal{A} \cup \mathcal{F}_\mathcal{C}$ of function symbols (including constant symbols),
- a set $\mathcal{V}_\mathcal{A}$ of regular variable symbols, and
- a set $\mathcal{V}_\mathcal{C} \subseteq \mathcal{T}(\mathcal{F}_\mathcal{A})$ of constraint variable symbols, where $\mathcal{T}(\mathcal{F}_\mathcal{A})$ denotes the set of all ground terms over $\mathcal{F}_\mathcal{A}$

As usual, the atoms in $\mathcal{A} \cup \mathcal{C}$ are obtained by a grounding process

Potassco

An Example

$$
\begin{aligned}
& time(0..t_{max}) \\
& bucket(a) \quad bucket(b) \\
1\ \{pour(B,T) : bucket(B)\}\ 1 \ &\leftarrow\ time(T), T < t_{max} \\
1 \leq^{\$} amt(B,T) \ &\leftarrow\ pour(B,T), T < t_{max} \\
amt(B,T) \leq^{\$} 3 \ &\leftarrow\ pour(B,T), T < t_{max} \\
amt(B,T) =^{\$} 0 \ &\leftarrow\ {\sim}pour(B,T), T < t_{max} \\
vol(B,T{+}1) =^{\$} vol(B,T) +^{\$} amt(B,T) \ &\leftarrow\ time(T), T < t_{max} \\
down(B,T) \ &\leftarrow\ vol(C,T) <^{\$} vol(B,T) \\
up(B,T) \ &\leftarrow\ {\sim}down(B,T) \\
vol(a,0) =^{\$} 0 \quad vol(b,0) =^{\$} 1 & \\
&\leftarrow\ up(a,t_{max})
\end{aligned}
$$

Potassco

An Example

- Consider the signature of our exemplary program:

$$\begin{aligned} \{B, C, T\} &\subseteq \mathcal{V}_{\mathcal{A}} \\ \{0, \ldots, t_{max}, +, a, b, amt, vol\} &\subseteq \mathcal{F}_{\mathcal{A}} \\ \{<, time, bucket, pour, up, down\} &\subseteq \mathcal{P}_{\mathcal{A}} \\ \{0, 1, 3, +^{\$}\} &\subseteq \mathcal{F}_{\mathcal{C}} \\ \{=^{\$}, <^{\$}, \leq^{\$}\} &\subseteq \mathcal{P}_{\mathcal{C}} \end{aligned}$$

- With substitution $\{B \mapsto b, T \mapsto 1, t_{max} \mapsto 2\}$, we get:

$$amt(b,1) =^{\$} 0 \leftarrow \sim pour(b,1)$$

$$vol(b,2) =^{\$} vol(b,1) +^{\$} amt(b,1) \leftarrow$$

and, among others, our signature hence contains

$$\begin{aligned} \{amt(b,1), vol(b,1), vol(b,2)\} &\subseteq \mathcal{V}_{\mathcal{C}} \\ \{pour(b,1)\} &\subseteq \mathcal{A} \\ \{amt(b,1) =^{\$} 0,\ vol(b,2) =^{\$} vol(b,1) +^{\$} amt(b,1)\} &\subseteq \mathcal{C} \end{aligned}$$

Potassco

An Example

- Consider the signature of our exemplary program:

$$\begin{aligned} \{B, C, T\} &\subseteq \mathcal{V}_{\mathcal{A}} \\ \{0, \ldots, t_{max}, +, a, b, amt, vol\} &\subseteq \mathcal{F}_{\mathcal{A}} \\ \{<, time, bucket, pour, up, down\} &\subseteq \mathcal{P}_{\mathcal{A}} \\ \{0, 1, 3, +^{\$}\} &\subseteq \mathcal{F}_{\mathcal{C}} \\ \{=^{\$}, <^{\$}, \leq^{\$}\} &\subseteq \mathcal{P}_{\mathcal{C}} \end{aligned}$$

- With substitution $\{B \mapsto b, T \mapsto 1, t_{max} \mapsto 2\}$, we get:

$$\begin{aligned} amt(b,1) =^{\$} 0 &\leftarrow \sim pour(b,1) \\ vol(b,2) =^{\$} vol(b,1) +^{\$} amt(b,1) &\leftarrow \end{aligned}$$

and, among others, our signature hence contains

$$\begin{aligned} \{amt(b,1), vol(b,1), vol(b,2)\} &\subseteq \mathcal{V}_{\mathcal{C}} \\ \{pour(b,1)\} &\subseteq \mathcal{A} \\ \{amt(b,1) =^{\$} 0,\ vol(b,2) =^{\$} vol(b,1) +^{\$} amt(b,1)\} &\subseteq \mathcal{C} \end{aligned}$$

Potassco

An Example

- Consider the signature of our exemplary program:

$$\begin{aligned} \{B, C, T\} &\subseteq \mathcal{V}_{\mathcal{A}} \\ \{0, \ldots, t_{max}, +, a, b, amt, vol\} &\subseteq \mathcal{F}_{\mathcal{A}} \\ \{<, time, bucket, pour, up, down\} &\subseteq \mathcal{P}_{\mathcal{A}} \\ \{0, 1, 3, +^{\$}\} &\subseteq \mathcal{F}_{\mathcal{C}} \\ \{=^{\$}, <^{\$}, \leq^{\$}\} &\subseteq \mathcal{P}_{\mathcal{C}} \end{aligned}$$

- With substitution $\{B \mapsto b, T \mapsto 1, t_{max} \mapsto 2\}$, we get:

$$\begin{aligned} amt(b,1) =^{\$} 0 &\leftarrow \sim pour(b,1) \\ vol(b,2) =^{\$} vol(b,1) +^{\$} amt(b,1) &\leftarrow \end{aligned}$$

and, among others, our signature hence contains

$$\begin{aligned} \{amt(b,1), vol(b,1), vol(b,2)\} &\subseteq \mathcal{V}_{\mathcal{C}} \\ \{pour(b,1)\} &\subseteq \mathcal{A} \\ \{amt(b,1) =^{\$} 0,\ vol(b,2) =^{\$} vol(b,1) +^{\$} amt(b,1)\} &\subseteq \mathcal{C} \end{aligned}$$

Potassco

An Example

- Consider the signature of our exemplary program:

$$\begin{aligned} \{B, C, T\} &\subseteq \mathcal{V}_{\mathcal{A}} \\ \{0, \ldots, t_{max}, +, a, b, amt, vol\} &\subseteq \mathcal{F}_{\mathcal{A}} \\ \{<, time, bucket, pour, up, down\} &\subseteq \mathcal{P}_{\mathcal{A}} \\ \{0, 1, 3, +^{\$}\} &\subseteq \mathcal{F}_{\mathcal{C}} \\ \{=^{\$}, <^{\$}, \leq^{\$}\} &\subseteq \mathcal{P}_{\mathcal{C}} \end{aligned}$$

- With substitution $\{B \mapsto b, T \mapsto 1, t_{max} \mapsto 2\}$, we get:

$$amt(b,1) =^{\$} 0 \leftarrow \sim pour(b,1)$$

$$vol(b,2) =^{\$} vol(b,1) +^{\$} amt(b,1) \leftarrow$$

and, among others, our signature hence contains

$$\begin{aligned} \{amt(b,1), vol(b,1), vol(b,2)\} &\subseteq \mathcal{V}_{\mathcal{C}} \\ \{pour(b,1)\} &\subseteq \mathcal{A} \\ \{amt(b,1) =^{\$} 0,\ vol(b,2) =^{\$} vol(b,1) +^{\$} amt(b,1)\} &\subseteq \mathcal{C} \end{aligned}$$

Potassco

An Example

- Consider the signature of our exemplary program:

$$\begin{aligned} \{B, C, T\} &\subseteq \mathcal{V}_{\mathcal{A}} \\ \{0, \ldots, t_{max}, +, a, b, amt, vol\} &\subseteq \mathcal{F}_{\mathcal{A}} \\ \{<, time, bucket, pour, up, down\} &\subseteq \mathcal{P}_{\mathcal{A}} \\ \{0, 1, 3, +^{\$}\} &\subseteq \mathcal{F}_{\mathcal{C}} \\ \{=^{\$}, <^{\$}, \leq^{\$}\} &\subseteq \mathcal{P}_{\mathcal{C}} \end{aligned}$$

- With substitution $\{B \mapsto b, T \mapsto 1, t_{max} \mapsto 2\}$, we get:

$$\begin{aligned} amt(b,1) =^{\$} 0 &\leftarrow \sim pour(b,1) \\ vol(b,2) =^{\$} vol(b,1) +^{\$} amt(b,1) &\leftarrow \end{aligned}$$

and, among others, our signature hence contains

$$\begin{aligned} \{amt(b,1), vol(b,1), vol(b,2)\} &\subseteq \mathcal{V}_{\mathcal{C}} \\ \{pour(b,1)\} &\subseteq \mathcal{A} \\ \{amt(b,1) =^{\$} 0,\ vol(b,2) =^{\$} vol(b,1) +^{\$} amt(b,1)\} &\subseteq \mathcal{C} \end{aligned}$$

Potassco

An Example

- For $t_{max} = 2$, our program has eleven constraint answer sets, summarized as follows

$up(a,0)$	$pour(a,0)$	$amt(a,0)$	$up(a,1)$	$pour(a,1)$	$amt(a,1)$	$up(a,2)$
T	**T**	1	**T**	**T**	1, 2, 3	**F**
T	**T**	2, 3	**F**	**T**	1, 2, 3	**F**
T	**T**	3	**F**	**F**	0	**F**
T	**F**	0	**T**	**T**	3	**F**

Potassco

An Example

- For $t_{max} = 2$, our program has eleven constraint answer sets, summarized as follows

$up(a,0)$	$pour(a,0)$	$amt(a,0)$	$up(a,1)$	$pour(a,1)$	$amt(a,1)$	$up(a,2)$
T	**T**	1	**T**	**T**	$1,2,3$	**F**
T	**T**	$2,3$	**F**	**T**	$1,2,3$	**F**
T	**T**	3	**F**	**F**	0	**F**
T	**F**	0	**T**	**T**	3	**F**

Potassco

Outline

Potassco

SAT modulo theories

SAT modulo theories (SMT)

- logical formulas with respect to combinations of background theories
- real numbers, integers, lists, arrays, bit vectors ...

Potassco

SAT modulo theories

SAT modulo theories (SMT)

- logical formulas with respect to combinations of background theories
- real numbers, integers, lists, arrays, bit vectors ...

SAT

$$(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$$

Potassco

SAT modulo theories

SAT modulo theories (SMT)

- logical formulas with respect to combinations of background theories
- real numbers, integers, lists, arrays, bit vectors ...

SMT

$$(\sin(x)^3 = \cos(\log(y) \cdot x) \quad \vee b \quad \vee -x^2 \geq 2.3y)$$

Potassco

SAT modulo theories

SAT modulo theories (SMT)

- logical formulas with respect to combinations of background theories
- real numbers, integers, lists, arrays, bit vectors ...

SMT

$$(\quad a \quad \vee b \quad \vee -x^2 \geq 2.3y)$$

Potassco

SAT modulo theories

SAT modulo theories (SMT)

- logical formulas with respect to combinations of background theories
- real numbers, integers, lists, arrays, bit vectors . . .

SMT

$$(\quad a \quad \vee b \quad \vee \quad c \quad)$$

Potassco

CDNL

Main Algorithm

loop

- Propagate
- **if** no conflict **then**
  - **if** partial assignment **then** Decide
    else return solution
- **else if** some decisions made **then**
  - Analyze Conflict
    Record Reason
    Backjump
- **else exit**

Potassco

CDNL

Main Algorithm

loop

- Propagate
- **if** no conflict **then**
  - **if** partial assignment **then** Decide
    else return solution
- **else if** some decisions made **then**
  - Analyze Conflict
    Record Reason
    Backjump
- **else exit**

Potassco

ASP Propagation

Propagation Algorithm

loop

UNIT-PROPAGATION

if conflict **then return**

else if UNFOUNDED-SET **then**

RECORD LOOP-NOGOOD

if conflict **then return**

else return

Potassco

Constraint ASP Propagation

Propagation Algorithm

loop

UNIT-PROPAGATION

if conflict **then return**

else if UNFOUNDED-SET **then**

RECORD LOOP-NOGOOD

if conflict **then return**

else if CONSTRAINT-PROPAGATION **then**

if conflict **then return**

else return

Potassco

CDNL

Main Algorithm

loop

- Propagate
- **if** no conflict **then**
  - **if** partial assignment **then** Decide
    else return solution
- **else if** some decisions made **then**
  - Analyze Conflict
    Record Reason
    Backjump
- **else exit**

Potassco

Constraint CDNL

Main Algorithm

loop

- Propagate
- **if** no conflict **then**
  - **if** partial assignment **then** Decide
  - **else if** CSP-Solve **then return** solution
  - **else**
    - Analyze Conflict
    - Record Reason
    - Backjump
- **else if** some decisions made **then**
  - Analyze Conflict
  - Record Reason
  - Backjump
- **else exit**

Conflict and reason determination

- Design Goal Use off-the-shelf CP solvers
- Problem Off-the-shelf CP solvers do not provide reasons for conflicts
- Idea Approximate reasons in various ways
  - lazy learning versus early learning
  - use constraint interdependencies or not

Potassco

Conflict and reason determination

- Design Goal Use off-the-shelf CP solvers
- Problem Off-the-shelf CP solvers do not provide reasons for conflicts
- Idea Approximate reasons in various ways
  - lazy learning versus early learning
  - use constraint interdependencies or not

Potassco

[1] C. Anger, M. Gebser, T. Linke, A. Neumann, and T. Schaub.
The nomore++ approach to answer set solving.
In G. Sutcliffe and A. Voronkov, editors, *Proceedings of the Twelfth International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'05)*, volume 3835 of *Lecture Notes in Artificial Intelligence*, pages 95–109. Springer-Verlag, 2005.

[2] C. Anger, K. Konczak, T. Linke, and T. Schaub.
A glimpse of answer set programming.
Künstliche Intelligenz, 19(1):12–17, 2005.

[3] Y. Babovich and V. Lifschitz.
Computing answer sets using program completion.
Unpublished draft, 2003.

[4] C. Baral.
Knowledge Representation, Reasoning and Declarative Problem Solving.
Cambridge University Press, 2003.

Potassco

[5] C. Baral, G. Brewka, and J. Schlipf, editors.
Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07), volume 4483 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2007.

[6] C. Baral and M. Gelfond.
Logic programming and knowledge representation.
Journal of Logic Programming, 12:1–80, 1994.

[7] S. Baselice, P. Bonatti, and M. Gelfond.
Towards an integration of answer set and constraint solving.
In M. Gabbrielli and G. Gupta, editors, *Proceedings of the Twenty-first International Conference on Logic Programming (ICLP'05)*, volume 3668 of *Lecture Notes in Computer Science*, pages 52–66. Springer-Verlag, 2005.

[8] A. Biere.
Adaptive restart strategies for conflict driven SAT solvers.

Potassco

In H. Kleine Büning and X. Zhao, editors, *Proceedings of the Eleventh International Conference on Theory and Applications of Satisfiability Testing (SAT'08)*, volume 4996 of *Lecture Notes in Computer Science*, pages 28–33. Springer-Verlag, 2008.

[9] A. Biere.
PicoSAT essentials.
Journal on Satisfiability, Boolean Modeling and Computation, 4:75–97, 2008.

[10] A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors.
Handbook of Satisfiability, volume 185 of *Frontiers in Artificial Intelligence and Applications*.
IOS Press, 2009.

[11] G. Brewka, T. Eiter, and M. Truszczyński.
Answer set programming at a glance.
Communications of the ACM, 54(12):92–103, 2011.

[12] K. Clark.
Negation as failure.

Potassco

In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, 1978.

[13] M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors. *Handbook of Tableau Methods*. Kluwer Academic Publishers, 1999.

[14] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. In *Proceedings of the Twelfth Annual IEEE Conference on Computational Complexity (CCC'97)*, pages 82–101. IEEE Computer Society Press, 1997.

[15] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5:394–397, 1962.

[16] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.

Potassco

[17] C. Drescher, M. Gebser, T. Grote, B. Kaufmann, A. König, M. Ostrowski, and T. Schaub.
Conflict-driven disjunctive answer set solving.
In G. Brewka and J. Lang, editors, *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*, pages 422–432. AAAI Press, 2008.

[18] C. Drescher, M. Gebser, B. Kaufmann, and T. Schaub.
Heuristics in conflict resolution.
In M. Pagnucco and M. Thielscher, editors, *Proceedings of the Twelfth International Workshop on Nonmonotonic Reasoning (NMR'08)*, number UNSW-CSE-TR-0819 in School of Computer Science and Engineering, The University of New South Wales, Technical Report Series, pages 141–149, 2008.

[19] N. Eén and N. Sörensson.
An extensible SAT-solver.
In E. Giunchiglia and A. Tacchella, editors, *Proceedings of the Sixth International Conference on Theory and Applications of Satisfiability*

Testing (SAT'03), volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer-Verlag, 2004.

[20] T. Eiter and G. Gottlob.
On the computational cost of disjunctive logic programming: Propositional case.
Annals of Mathematics and Artificial Intelligence, 15(3-4):289–323, 1995.

[21] T. Eiter, G. Ianni, and T. Krennwallner.
Answer Set Programming: A Primer.
In S. Tessaris, E. Franconi, T. Eiter, C. Gutierrez, S. Handschuh, M. Rousset, and R. Schmidt, editors, *Fifth International Reasoning Web Summer School (RW'09)*, volume 5689 of *Lecture Notes in Computer Science*, pages 40–110. Springer-Verlag, 2009.

[22] F. Fages.
Consistency of Clark's completion and the existence of stable models.
Journal of Methods of Logic in Computer Science, 1:51–60, 1994.

[23] P. Ferraris.

Potassco

Answer sets for propositional theories.
In C. Baral, G. Greco, N. Leone, and G. Terracina, editors, *Proceedings of the Eighth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'05)*, volume 3662 of *Lecture Notes in Artificial Intelligence*, pages 119–131. Springer-Verlag, 2005.

[24] P. Ferraris and V. Lifschitz.
Mathematical foundations of answer set programming.
In S. Artëmov, H. Barringer, A. d'Avila Garcez, L. Lamb, and J. Woods, editors, *We Will Show Them! Essays in Honour of Dov Gabbay*, volume 1, pages 615–664. College Publications, 2005.

[25] M. Fitting.
A Kripke-Kleene semantics for logic programs.
Journal of Logic Programming, 2(4):295–312, 1985.

[26] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and S. Thiele.
A user's guide to `gringo`, `clasp`, `clingo`, and `iclingo`.

Potassco

[27] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and S. Thiele.
Engineering an incremental ASP solver.
In M. Garcia de la Banda and E. Pontelli, editors, *Proceedings of the Twenty-fourth International Conference on Logic Programming (ICLP'08)*, volume 5366 of *Lecture Notes in Computer Science*, pages 190–205. Springer-Verlag, 2008.

[28] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub.
On the implementation of weight constraint rules in conflict-driven ASP solvers.
In Hill and Warren [44], pages 250–264.

[29] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub.
Answer Set Solving in Practice.
Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2012.

[30] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub.

Potassco

clasp: A conflict-driven answer set solver.
In Baral et al. [5], pages 260–265.

[31] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub.
Conflict-driven answer set enumeration.
In Baral et al. [5], pages 136–148.

[32] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub.
Conflict-driven answer set solving.
In Veloso [68], pages 386–392.

[33] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub.
Advanced preprocessing for answer set solving.
In M. Ghallab, C. Spyropoulos, N. Fakotakis, and N. Avouris, editors, *Proceedings of the Eighteenth European Conference on Artificial Intelligence (ECAI'08)*, pages 15–19. IOS Press, 2008.

[34] M. Gebser, B. Kaufmann, and T. Schaub.
The conflict-driven answer set solver clasp: Progress report.

Potassco

In E. Erdem, F. Lin, and T. Schaub, editors, *Proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09)*, volume 5753 of *Lecture Notes in Artificial Intelligence*, pages 509–514. Springer-Verlag, 2009.

[35] M. Gebser, B. Kaufmann, and T. Schaub.
Solution enumeration for projected Boolean search problems.
In W. van Hoeve and J. Hooker, editors, *Proceedings of the Sixth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'09)*, volume 5547 of *Lecture Notes in Computer Science*, pages 71–86. Springer-Verlag, 2009.

[36] M. Gebser, M. Ostrowski, and T. Schaub.
Constraint answer set solving.
In Hill and Warren [44], pages 235–249.

[37] M. Gebser and T. Schaub.
Tableau calculi for answer set programming.

Potassco

In S. Etalle and M. Truszczyński, editors, *Proceedings of the Twenty-second International Conference on Logic Programming (ICLP'06)*, volume 4079 of *Lecture Notes in Computer Science*, pages 11–25. Springer-Verlag, 2006.

[38] M. Gebser and T. Schaub.
Generic tableaux for answer set programming.
In V. Dahl and I. Niemelä, editors, *Proceedings of the Twenty-third International Conference on Logic Programming (ICLP'07)*, volume 4670 of *Lecture Notes in Computer Science*, pages 119–133. Springer-Verlag, 2007.

[39] M. Gelfond.
Answer sets.
In V. Lifschitz, F. van Harmelen, and B. Porter, editors, *Handbook of Knowledge Representation*, chapter 7, pages 285–316. Elsevier Science, 2008.

[40] M. Gelfond and N. Leone.

Potassco

Logic programming and knowledge representation — the A-Prolog perspective.
Artificial Intelligence, 138(1-2):3–38, 2002.

[41] M. Gelfond and V. Lifschitz.
The stable model semantics for logic programming.
In R. Kowalski and K. Bowen, editors, *Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88)*, pages 1070–1080. MIT Press, 1988.

[42] M. Gelfond and V. Lifschitz.
Logic programs with classical negation.
In D. Warren and P. Szeredi, editors, *Proceedings of the Seventh International Conference on Logic Programming (ICLP'90)*, pages 579–597. MIT Press, 1990.

[43] E. Giunchiglia, Y. Lierler, and M. Maratea.
Answer set programming based on propositional satisfiability.
Journal of Automated Reasoning, 36(4):345–377, 2006.

Potassco

[44] P. Hill and D. Warren, editors.
Proceedings of the Twenty-fifth International Conference on Logic Programming (ICLP'09), volume 5649 of *Lecture Notes in Computer Science*. Springer-Verlag, 2009.

[45] J. Huang.
The effect of restarts on the efficiency of clause learning.
In Veloso [68], pages 2318–2323.

[46] K. Konczak, T. Linke, and T. Schaub.
Graphs and colorings for answer set programming.
Theory and Practice of Logic Programming, 6(1-2):61–106, 2006.

[47] J. Lee.
A model-theoretic counterpart of loop formulas.
In L. Kaelbling and A. Saffiotti, editors, *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 503–508. Professional Book Center, 2005.

Potassco

[48] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello.
The DLV system for knowledge representation and reasoning.
ACM Transactions on Computational Logic, 7(3):499–562, 2006.

[49] V. Lifschitz.
Answer set programming and plan generation.
Artificial Intelligence, 138(1-2):39–54, 2002.

[50] V. Lifschitz.
Introduction to answer set programming.
Unpublished draft, 2004.

[51] V. Lifschitz and A. Razborov.
Why are there so many loop formulas?
ACM Transactions on Computational Logic, 7(2):261–268, 2006.

[52] F. Lin and Y. Zhao.
ASSAT: computing answer sets of a logic program by SAT solvers.
Artificial Intelligence, 157(1-2):115–137, 2004.

Potassco

[53] V. Marek and M. Truszczyński.
Nonmonotonic logic: context-dependent reasoning.
Artifical Intelligence. Springer-Verlag, 1993.

[54] V. Marek and M. Truszczyński.
Stable models and an alternative logic programming paradigm.
In K. Apt, V. Marek, M. Truszczyński, and D. Warren, editors, *The Logic Programming Paradigm: a 25-Year Perspective*, pages 375–398. Springer-Verlag, 1999.

[55] J. Marques-Silva, I. Lynce, and S. Malik.
Conflict-driven clause learning SAT solvers.
In Biere et al. [10], chapter 4, pages 131–153.

[56] J. Marques-Silva and K. Sakallah.
GRASP: A search algorithm for propositional satisfiability.
IEEE Transactions on Computers, 48(5):506–521, 1999.

[57] V. Mellarkod and M. Gelfond.
Integrating answer set reasoning with constraint solving techniques.

In J. Garrigue and M. Hermenegildo, editors, *Proceedings of the Ninth International Symposium on Functional and Logic Programming (FLOPS'08)*, volume 4989 of *Lecture Notes in Computer Science*, pages 15–31. Springer-Verlag, 2008.

[58] V. Mellarkod, M. Gelfond, and Y. Zhang.
Integrating answer set programming and constraint logic programming.
Annals of Mathematics and Artificial Intelligence, 53(1-4):251–287, 2008.

[59] D. Mitchell.
A SAT solver primer.
Bulletin of the European Association for Theoretical Computer Science, 85:112–133, 2005.

[60] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik.
Chaff: Engineering an efficient SAT solver.
In *Proceedings of the Thirty-eighth Conference on Design Automation (DAC'01)*, pages 530–535. ACM Press, 2001.

Potassco

[61] I. Niemelä.
Logic programs with stable model semantics as a constraint programming paradigm.
Annals of Mathematics and Artificial Intelligence, 25(3-4):241–273, 1999.

[62] R. Nieuwenhuis, A. Oliveras, and C. Tinelli.
Solving SAT and SAT modulo theories: From an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T).
Journal of the ACM, 53(6):937–977, 2006.

[63] K. Pipatsrisawat and A. Darwiche.
A lightweight component caching scheme for satisfiability solvers.
In J. Marques-Silva and K. Sakallah, editors, *Proceedings of the Tenth International Conference on Theory and Applications of Satisfiability Testing (SAT'07)*, volume 4501 of *Lecture Notes in Computer Science*, pages 294–299. Springer-Verlag, 2007.

[64] L. Ryan.
Efficient algorithms for clause-learning SAT solvers.

Potassco

Master's thesis, Simon Fraser University, 2004.

[65] P. Simons, I. Niemelä, and T. Soininen.
Extending and implementing the stable model semantics.
Artificial Intelligence, 138(1-2):181–234, 2002.

[66] T. Syrjänen.
Lparse 1.0 user's manual.

[67] A. Van Gelder, K. Ross, and J. Schlipf.
The well-founded semantics for general logic programs.
Journal of the ACM, 38(3):620–650, 1991.

[68] M. Veloso, editor.
Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07). AAAI/MIT Press, 2007.

[69] L. Zhang, C. Madigan, M. Moskewicz, and S. Malik.
Efficient conflict driven learning in a Boolean satisfiability solver.
In *Proceedings of the International Conference on Computer-Aided Design (ICCAD'01)*, pages 279–285. ACM Press, 2001.

Potassco