Answer Set Programming
Theory, Practice, and Beyond

Torsten Schaub

University of Potsdam

Potassco

ASP and SAT

- SAT = ASP + Excluded middle formulas
- ASP = SAT + Completion and Loop formulas

- Note Checking whether a propositional formula has a stable model is Σ_P^2-complete

Potassco

ASP and SAT

- SAT = ASP + Excluded middle formulas
- ASP = SAT + Completion and Loop formulas

- Note Checking whether a propositional formula has a stable model is Σ_P^2-complete

Potassco

ASP and SAT

- SAT = ASP + Excluded middle formulas [1]
- ASP = SAT + Completion and Loop formulas

- Note Checking whether a propositional formula has a stable model is Σ^2_P-complete

[1] For instance, '{a}.' stands for '$a \vee \neg a$'.

Potassco

ASP and SAT

- SAT = ASP + Excluded middle formulas
- ASP = SAT + Completion and Loop formulas

- Note Checking whether a propositional formula has a stable model is Σ_P^2-complete

Potassco

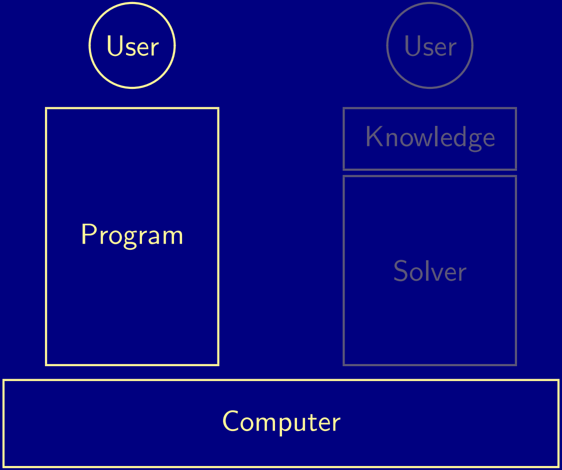
Outline

Potassco

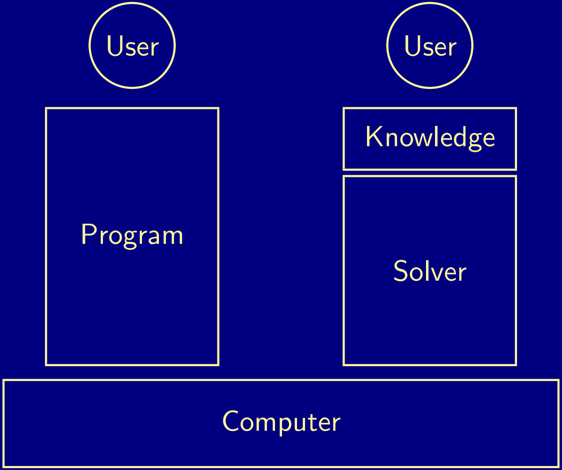
Outline

Potassco

Traditional Software



Knowledge-driven Software



Potassco

What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability

Knowledge **Expert**

Solver

Potassco

What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability

Knowledge **Expert**

Solver

Potassco

What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability

Knowledge **Expert**

Solver

Industrial impact

Within SIEMENS, constraint technologies have been successfully used for solving configuration problems for more than 25 years. [...] approximately 80 percent of the maintenance costs and more than 60 percent of the development costs for the knowledge representation and reasoning tasks were saved.

In: A. Falkner et al. Twenty-Five Years of Successful Application of Constraint Technologies at Siemens. AI Magazine. 37(4):67-80, 2016.

Potassco

Industrial impact

Within SIEMENS, constraint technologies have been successfully used for solving configuration problems for more than 25 years. [...] approximately 80 percent of the maintenance costs and more than 60 percent of the development costs for the knowledge representation and reasoning tasks were saved.

In: A. Falkner et al. Twenty-Five Years of Successful Application of Constraint Technologies at Siemens. AI Magazine. 37(4):67-80, 2016.

Potassco

Industrial impact

Within SIEMENS, constraint technologies have been successfully used for solving configuration problems for more than 25 years. [...] approximately 80 percent of the maintenance costs and more than 60 percent of the development costs for the knowledge representation and reasoning tasks were saved.

In: A. Falkner et al. Twenty-Five Years of Successful Application of Constraint Technologies at Siemens. AI Magazine. 37(4):67-80, 2016.

Potassco

Outline

Potassco

Answer Set Programming

in a Walnutshell

- ASP is an approach to declarative problem solving, featuring
  - a rich yet simple modeling language
  - high-performance solving capacities
  - closed and open world reasoning
  - qualitative and quantitative optimization

  tailored to Knowledge Representation and Reasoning
- ASP allows for solving all search problems in NP (and NP^{NP}) in a uniform way

Potassco

Answer Set Programming

in a Walnutshell

- ASP is an approach to declarative problem solving, featuring
  - a rich yet simple modeling language
  - high-performance solving capacities
  - closed and open world reasoning
  - qualitative and quantitative optimization

  tailored to Knowledge Representation and Reasoning
- ASP allows for solving all search problems in NP (and NP^{NP}) in a uniform way

Potassco

Answer Set Programming

in a Walnutshell

- ASP is an approach to declarative problem solving, featuring
  - a rich yet simple modeling language
  - high-performance solving capacities
  - closed and open world reasoning
  - qualitative and quantitative optimization

  tailored to Knowledge Representation and Reasoning
- ASP allows for solving all search problems in NP (and NP^{NP}) in a uniform way

$$\textbf{ASP} = \textbf{DB}+\textbf{LP}+\textbf{KR}+\textbf{SAT}$$

Outline

Open and Closed world reasoning

- Closed world reasoning
  - if a statement is true, it remains true
  - if a statement is false, it remains false
  - if a statement is unknown, it becomes false

- Open world reasoning
  - if a statement is true, it remains true
  - if a statement is false, it remains false
  - if a statement is unknown, it is either true or false

Potassco

Open and Closed world reasoning

- Closed world reasoning
  - if a statement is true, it remains true
  - if a statement is false, it remains false
  - if a statement is unknown, it becomes false
- Open world reasoning
  - if a statement is true, it remains true
  - if a statement is false, it remains false
  - if a statement is unknown, it is either true or false

Open and Closed world reasoning

- Closed world reasoning
  - if a statement is true, it remains true
  - if a statement is false, it remains false
  - if a statement is unknown, it becomes false

  is non-monotonic

- Open world reasoning
  - if a statement is true, it remains true
  - if a statement is false, it remains false
  - if a statement is unknown, it is either true or false

  is monotonic

Open and Closed world reasoning

- Closed world reasoning
  - if a statement is true, it remains true
  - if a statement is false, it remains false
  - if a statement is unknown, it becomes false

  is non-monotonic

  offers defaults, succinctness
- Open world reasoning
  - if a statement is true, it remains true
  - if a statement is false, it remains false
  - if a statement is unknown, it is either true or false

  is monotonic

Open and Closed world reasoning

- Closed world reasoning
  - if a statement is true, it remains true
  - if a statement is false, it remains false
  - if a statement is unknown, it becomes false

  is non-monotonic

  offers defaults, succinctness
- Open world reasoning
  - if a statement is true, it remains true
  - if a statement is false, it remains false
  - if a statement is unknown, it is either true or false

  is monotonic

- ASP offers both open and closed world reasoning
  by using stable model semantics

Open and Closed world reasoning

by example

- Alphabet $\{a, b\}$
- The rule
  - a

  has the
  - models $\{a\}, \{a, b\}$
  - minimal models $\{a\}$
  - stable models $\{a\}$

Potassco

Open and Closed world reasoning

by example

- Alphabet $\{a, b\}$
- The fact
  - a

  has the
  - models $\{a\}, \{a, b\}$
  - minimal models $\{a\}$
  - stable models $\{a\}$

Open and Closed world reasoning

by example

- Alphabet $\{a, b\}$
- The rule
  - $\neg b \rightarrow a$

  has the
  - models $\{a\}, \{b\}, \{a, b\}$
  - minimal models $\{a\}, \{b\}$
  - stable models $\{a\}$

Potassco

The logic of Here-and-There (HT)

- Formula $\varphi \;::=\; \bot \mid a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi$
- Interpretation A pair $\langle H, T\rangle$ of sets of atoms with $H \subseteq T$
  - H is called "here" and
  - T is called "there"
- Note $\langle H, T\rangle$ is a simplified Kripke structure
- Intuition
  - H represents provably true atoms
  - T represents possibly true atoms
  - atoms not in T are false
- Idea
  - $\langle H, T\rangle \models \varphi \quad \sim \quad \varphi$ is provably true
  - $\langle T, T\rangle \models \varphi \quad \sim \quad \varphi$ is possibly true (ie, classically true)

Potassco

The logic of Here-and-There (HT)

- Formula $\varphi \ ::= \ \bot \ | \ a \ | \ \varphi \wedge \varphi \ | \ \varphi \vee \varphi \ | \ \varphi \rightarrow \varphi$
- Interpretation A pair $\langle H, T \rangle$ of sets of atoms with $H \subseteq T$
  - H is called "here" and
  - T is called "there"
- Note $\langle H, T \rangle$ is a simplified Kripke structure
- Intuition
  - H represents provably true atoms
  - T represents possibly true atoms
  - atoms not in T are false
- Idea
  - $\langle H, T \rangle \models \varphi \quad \sim \quad \varphi$ is provably true
  - $\langle T, T \rangle \models \varphi \quad \sim \quad \varphi$ is possibly true (ie, classically true)

Potassco

The logic of Here-and-There (HT)

- Formula $\varphi \ ::= \ \bot \ | \ a \ | \ \varphi \wedge \varphi \ | \ \varphi \vee \varphi \ | \ \varphi \rightarrow \varphi$
- Interpretation A pair $\langle H, T \rangle$ of sets of atoms with $H \subseteq T$
  - H is called "here" and
  - T is called "there"
- Note $\langle H, T \rangle$ is a simplified Kripke structure
- Intuition
  - H represents provably true atoms
  - T represents possibly true atoms
  - atoms not in T are false
- Idea
  - $\langle H, T \rangle \models \varphi \quad \sim \quad \varphi$ is provably true
  - $\langle T, T \rangle \models \varphi \quad \sim \quad \varphi$ is possibly true (ie, classically true)

Potassco

The logic of Here-and-There (HT)

- Formula $\varphi \ ::= \ \bot \ | \ a \ | \ \varphi \wedge \varphi \ | \ \varphi \vee \varphi \ | \ \varphi \rightarrow \varphi$
- Interpretation A pair $\langle H, T \rangle$ of sets of atoms with $H \subseteq T$
  - H is called "here" and
  - T is called "there"
- Note $\langle H, T \rangle$ is a simplified Kripke structure
- Intuition
  - H represents provably true atoms
  - T represents possibly true atoms
  - atoms not in T are false
- Idea
  - $\langle H, T \rangle \models \varphi \quad \sim \quad \varphi$ is provably true
  - $\langle T, T \rangle \models \varphi \quad \sim \quad \varphi$ is possibly true (ie, classically true)

Potassco

The logic of Here-and-There (HT)

- Formula $\varphi \;::=\; \bot \mid a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi$
- Interpretation A pair $\langle H, T \rangle$ of sets of atoms with $H \subseteq T$
  - H is called "here" and
  - T is called "there"
- Note $\langle H, T \rangle$ is a simplified Kripke structure
- Intuition
  - H represents provably true atoms
  - T represents possibly true atoms
  - atoms not in T are false
- Idea
  - $\langle H, T \rangle \models \varphi \quad \sim \quad \varphi$ is provably true
  - $\langle T, T \rangle \models \varphi \quad \sim \quad \varphi$ is possibly true (ie, classically true)

The logic of Here-and-There (HT)

- Formula $\varphi \ ::= \ \bot \ \mid \ a \ \mid \ \varphi \wedge \varphi \ \mid \ \varphi \vee \varphi \ \mid \ \varphi \rightarrow \varphi$
- Interpretation A pair $\langle H, T \rangle$ of sets of atoms with $H \subseteq T$
  - H is called "here" and
  - T is called "there"
- Note $\langle H, T \rangle$ is a simplified Kripke structure
- Intuition
  - H represents provably true atoms
  - T represents possibly true atoms
  - atoms not in T are false
- Idea
  - $\langle H, T \rangle \models \varphi \quad \sim \quad \varphi$ is provably true
  - $\langle T, T \rangle \models \varphi \quad \sim \quad \varphi$ is possibly true (ie, classically true)

Satisfaction

- $\langle H, T\rangle \models a$ if $a \in H$ for any atom a
- $\langle H, T\rangle \models \varphi \wedge \psi$ if $\langle H, T\rangle \models \varphi$ and $\langle H, T\rangle \models \psi$
- $\langle H, T\rangle \models \varphi \vee \psi$ if $\langle H, T\rangle \models \varphi$ or $\langle H, T\rangle \models \psi$
- $\langle H, T\rangle \models \varphi \rightarrow \psi$ if $\langle X, T\rangle \models \varphi$ implies $\langle X, T\rangle \models \psi$ for both $X = H, T$

- Note $\langle H, T\rangle \models \neg\varphi$ if $\langle T, T\rangle \not\models \varphi$ since $\neg\varphi = \varphi \rightarrow \bot$

- An interpretation $\langle H, T\rangle$ is a model of φ, if $\langle H, T\rangle \models \varphi$

Satisfaction

- $\langle H, T \rangle \models a$ if $a \in H$ for any atom a
- $\langle H, T \rangle \models \varphi \wedge \psi$ if $\langle H, T \rangle \models \varphi$ and $\langle H, T \rangle \models \psi$
- $\langle H, T \rangle \models \varphi \vee \psi$ if $\langle H, T \rangle \models \varphi$ or $\langle H, T \rangle \models \psi$
- $\langle H, T \rangle \models \varphi \rightarrow \psi$ if $\langle X, T \rangle \models \varphi$ implies $\langle X, T \rangle \models \psi$ for both $X = H, T$
- Note $\langle H, T \rangle \models \neg\varphi$ if $\langle T, T \rangle \not\models \varphi$ since $\neg\varphi = \varphi \rightarrow \bot$
- An interpretation $\langle H, T \rangle$ is a model of φ, if $\langle H, T \rangle \models \varphi$

Potassco

Satisfaction

- $\langle H, T \rangle \models a$ if $a \in H$ for any atom a
- $\langle H, T \rangle \models \varphi \wedge \psi$ if $\langle H, T \rangle \models \varphi$ and $\langle H, T \rangle \models \psi$
- $\langle H, T \rangle \models \varphi \vee \psi$ if $\langle H, T \rangle \models \varphi$ or $\langle H, T \rangle \models \psi$
- $\langle H, T \rangle \models \varphi \rightarrow \psi$ if $\langle X, T \rangle \models \varphi$ implies $\langle X, T \rangle \models \psi$ for both $X = H, T$

- Note $\langle H, T \rangle \models \neg\varphi$ if $\langle T, T \rangle \not\models \varphi$ since $\neg\varphi = \varphi \rightarrow \bot$

- An interpretation $\langle H, T \rangle$ is a model of φ, if $\langle H, T \rangle \models \varphi$

Satisfaction

- $\langle H, T \rangle \models a$ if $a \in H$ for any atom a
- $\langle H, T \rangle \models \varphi \wedge \psi$ if $\langle H, T \rangle \models \varphi$ and $\langle H, T \rangle \models \psi$
- $\langle H, T \rangle \models \varphi \vee \psi$ if $\langle H, T \rangle \models \varphi$ or $\langle H, T \rangle \models \psi$
- $\langle H, T \rangle \models \varphi \rightarrow \psi$ if $\langle X, T \rangle \models \varphi$ implies $\langle X, T \rangle \models \psi$
  for both $X = H, T$

- Note $\langle H, T \rangle \models \neg\varphi$ if $\langle T, T \rangle \not\models \varphi$ since $\neg\varphi = \varphi \rightarrow \bot$

- An interpretation $\langle H, T \rangle$ is a model of φ, if $\langle H, T \rangle \models \varphi$

Classical tautologies

H	T	a	$\neg a$	$a \vee \neg a$	$\neg\neg a$	$\neg\neg a \vee \neg a$	$a \leftarrow \neg\neg a$
$\{a\}$	$\{a\}$	**T**	**F**	**T**	**T**	**T**	**T**
$\emptyset$	$\{a\}$	**F**	**F**	**F**	**T**	**T**	**F**
$\emptyset$	$\emptyset$	**F**	**T**	**T**	**F**	**T**	**T**

Potassco

Classical tautologies

H	T	a	$\neg a$	$a \vee \neg a$	$\neg\neg a$	$\neg\neg a \vee \neg a$	$a \leftarrow \neg\neg a$
$\{a\}$	$\{a\}$	**T**	**F**	**T**	**T**	**T**	**T**
$\emptyset$	$\{a\}$	**F**	**F**	**F**	**T**	**T**	**F**
$\emptyset$	$\emptyset$	**F**	**T**	**T**	**F**	**T**	**T**

Potassco

Equilibrium models

- A total interpretation $\langle T, T \rangle$ is an equilibrium model of a formula φ, if
  1. $\langle T, T \rangle \models \varphi$
  2. $\langle H, T \rangle \not\models \varphi$ for all $H \subset T$
- T is called a stable model of φ
- Note $\langle T, T \rangle$ acts as a classical model
- Note $\langle H, T \rangle \models P$ iff $H \models P^T$ (P^T is the reduct of P by T)

Potassco

Equilibrium models

- A total interpretation $\langle T, T \rangle$ is an equilibrium model of a formula φ, if
  1. $\langle T, T \rangle \models \varphi$
  2. $\langle H, T \rangle \not\models \varphi$ for all $H \subset T$
- T is called a stable model of φ
- Note $\langle T, T \rangle$ acts as a classical model
- Note $\langle H, T \rangle \models P$ iff $H \models P^T$ (P^T is the reduct of P by T)

Potassco

Equilibrium models

- A total interpretation $\langle T, T\rangle$ is an equilibrium model of a formula φ, if
  1. $\langle T, T\rangle \models \varphi$
  2. $\langle H, T\rangle \not\models \varphi$ for all $H \subset T$
- T is called a stable model of φ
- Note $\langle T, T\rangle$ acts as a classical model
- Note $\langle H, T\rangle \models P$ iff $H \models P^T$ (P^T is the reduct of P by T)

Potassco

Equilibrium models

- A total interpretation $\langle T, T \rangle$ is an equilibrium model of a formula φ, if
  1. $\langle T, T \rangle \models \varphi$
  2. $\langle H, T \rangle \not\models \varphi$ for all $H \subset T$
- T is called a stable model of φ

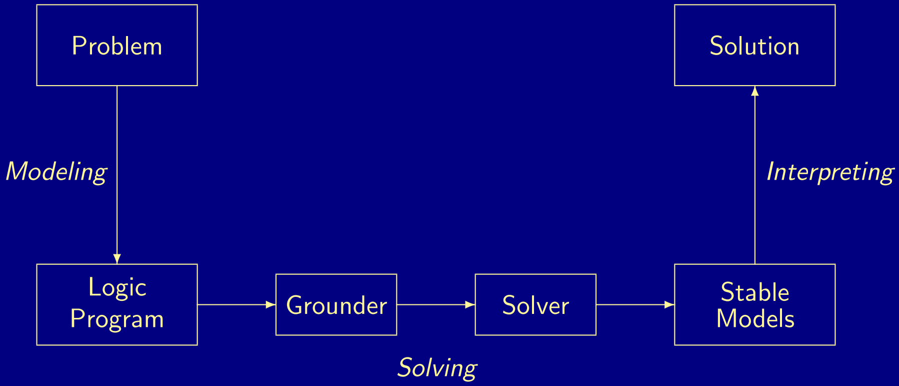
- Note $\langle T, T \rangle$ acts as a classical model
- Note $\langle H, T \rangle \models P$ iff $H \models P^T$ (P^T is the reduct of P by T)

Potassco

Outline

Potassco

Modeling, grounding, and solving



Language constructs

- Facts `q(42).`
- Rules `p(X) :- q(X), not r(X).`
- Conditional literals `p :- q(X) : r(X).`
- Disjunction `p(X) ; q(X) :- r(X).`
- Integrity constraints `:- q(X), p(X).`
- Choice `2 { p(X,Y) : q(X) } 7 :- r(Y).`
- Aggregates `s(Y) :- r(Y), 2 #sum{ X : p(X,Y), q(X) } 7.`
- Multi-objective optimization `:~ q(X), p(X,C). [C]`

  `#minimize { C : q(X), p(X,C) }`

Potassco

The traveling salesperson problem (TSP)

- Problem Instance A set of cities and distances among them, or simply a weighted graph
- Problem Class What is the shortest possible route visiting each city once and returning to the city of origin?
- Note
  - TSP extends the Hamiltonian cycle problem: Is there a cycle in a graph visiting each node exactly once
  - TSP is relevant to applications in logistics, planning, chip design, and the core of the vehicle routing problem

Potassco

The traveling salesperson problem (TSP)

- Problem Instance A set of cities and distances among them, or simply a weighted graph
- Problem Class What is the shortest possible route visiting each city once and returning to the city of origin?
- Note
  - TSP extends the Hamiltonian cycle problem:
    Is there a cycle in a graph visiting each node exactly once
  - TSP is relevant to applications in logistics, planning, chip design, and the core of the vehicle routing problem

Potassco

Traveling salesperson

Problem instance, cities.lp

```
start(a).

city(a). city(b). city(c). city(d).

road(a,b,10). road(b,c,20). road(c,d,25). road(d,a,40).
road(b,d,30). road(d,c,25). road(c,a,35).
```

Traveling salesperson

Problem encoding, `tsp.lp`

```
{ travel(X,Y) } :- road(X,Y,_).

visited(Y) :- travel(X,Y), start(X).
visited(Y) :- travel(X,Y), visited(X).

:- city(X), not visited(X).

:- city(X), 2 { travel(X,Y) }.
:- city(X), 2 { travel(Y,X) }.
```

Traveling salesperson

Problem encoding, tsp.lp

```
{ travel(X,Y) } :- road(X,Y,_).

visited(Y) :- travel(X,Y), start(X).
visited(Y) :- travel(X,Y), visited(X).

:- city(X), not visited(X).

:- city(X), 2 { travel(X,Y) }.
:- city(X), 2 { travel(Y,X) }.

:~ travel(X,Y), road(X,Y,D). [D,X,Y]
```

Traveling salesperson

Problem encoding, `tsp.lp`

```
{ travel(X,Y) } :- road(X,Y,_).

visited(Y) :- travel(X,Y), start(X).
visited(Y) :- travel(X,Y), visited(X).

:- city(X), not visited(X).

:- city(X), 2 { travel(X,Y) }.
:- city(X), 2 { travel(Y,X) }.

#minimize { D,X,Y : travel(X,Y), road(X,Y,D) }.
```

Running salesperson

```
$ clingo tsp.lp cities.lp
clingo version 5.3.1
Reading...
Solving...
Answer: 1
start(a) [...] road(c,a,35)
travel(a,b) travel(b,d) travel(d,c) travel(c,a)
visited(b) visited(c) visited(d) visited(a)
Optimization: 100
Answer: 2
start(a) [...] road(c,a,35)
travel(a,b) travel(b,c) travel(c,d) travel(d,a)
visited(b) visited(c) visited(d) visited(a)
Optimization: 95
OPTIMUM FOUND

Models       : 2
  Optimum    : yes
Optimization : 95
Calls        : 1
Time         : 0.005s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.002s
```

Running salesperson

```
$ clingo tsp.lp cities.lp
clingo version 5.3.1
Reading...
Solving...
Answer: 1
start(a) [...] road(c,a,35)
travel(a,b) travel(b,d) travel(d,c) travel(c,a)
visited(b) visited(c) visited(d) visited(a)
Optimization: 100
Answer: 2
start(a) [...] road(c,a,35)
travel(a,b) travel(b,c) travel(c,d) travel(d,a)
visited(b) visited(c) visited(d) visited(a)
Optimization: 95
OPTIMUM FOUND

Models       : 2
  Optimum    : yes
Optimization : 95
Calls        : 1
Time         : 0.005s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.002s
```

Running salesperson

```
$ clingo tsp.lp cities.lp
clingo version 5.3.1
Reading...
Solving...
Answer: 1
start(a) [...] road(c,a,35)
travel(a,b) travel(b,d) travel(d,c) travel(c,a)
visited(b) visited(c) visited(d) visited(a)
Optimization: 100
Answer: 2
start(a) [...] road(c,a,35)
travel(a,b) travel(b,c) travel(c,d) travel(d,a)
visited(b) visited(c) visited(d) visited(a)
Optimization: 95
OPTIMUM FOUND

Models       : 2
  Optimum    : yes
Optimization : 95
Calls        : 1
Time         : 0.005s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.002s
```

Running salesperson

```
$ clingo tsp.lp cities.lp
clingo version 5.3.1
Reading...
Solving...
Answer: 1
start(a) [...] road(c,a,35)
travel(a,b) travel(b,d) travel(d,c) travel(c,a)
visited(b) visited(c) visited(d) visited(a)
Optimization: 100
Answer: 2
start(a) [...] road(c,a,35)
travel(a,b) travel(b,c) travel(c,d) travel(d,a)
visited(b) visited(c) visited(d) visited(a)
Optimization: 95
OPTIMUM FOUND

Models       : 2
  Optimum    : yes
Optimization : 95
Calls        : 1
Time         : 0.005s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.002s
```

Running salesperson

```
$ clingo tsp.lp cities.lp
clingo version 5.3.1
Reading...
Solving...
Answer: 1
start(a) [...] road(c,a,35)
travel(a,b) travel(b,d) travel(d,c) travel(c,a)
visited(b) visited(c) visited(d) visited(a)
Optimization: 100
Answer: 2
start(a) [...] road(c,a,35)
travel(a,b) travel(b,c) travel(c,d) travel(d,a)
visited(b) visited(c) visited(d) visited(a)
Optimization: 95
OPTIMUM FOUND

Models       : 2
  Optimum    : yes
Optimization : 95
Calls        : 1
Time         : 0.005s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.002s
```

Traveling salesperson

Alternative problem encoding

```
{ travel(X,Y) : road(X,Y,_) } = 1 :- city(X).
{ travel(X,Y) : road(X,Y,_) } = 1 :- city(Y).

visited(Y) :- travel(X,Y), start(X).
visited(Y) :- travel(X,Y), visited(X).

:- city(X), not visited(X).

#minimize { D,X,Y : travel(X,Y), road(X,Y,D) }.
```

Outline

Potassco

Motivation

- Increasing railway traffic demands global and flexible ways for scheduling trains in order to use railway networks to capacity
- Difficulty arises from dependencies among trains induced by connections and shared resources
- Train scheduling combines three distinct tasks
  - Routing
  - Conflict detection and resolution
  - Scheduling
- Solution operational at Swiss Federal Railway using *clingo*[DL]
  - ASP
  - Difference constraints
  - (Hybrid) Optimization
  - Heuristic directives
  - Multi-shot solving

Potassco

Motivation

- Increasing railway traffic demands global and flexible ways for scheduling trains in order to use railway networks to capacity
- Difficulty arises from dependencies among trains induced by connections and shared resources
- Train scheduling combines three distinct tasks
  - Routing
  - Conflict detection and resolution
  - Scheduling
- Solution operational at Swiss Federal Railway using *clingo*[DL]
  - ASP
  - Difference constraints
  - (Hybrid) Optimization
  - Heuristic directives
  - Multi-shot solving

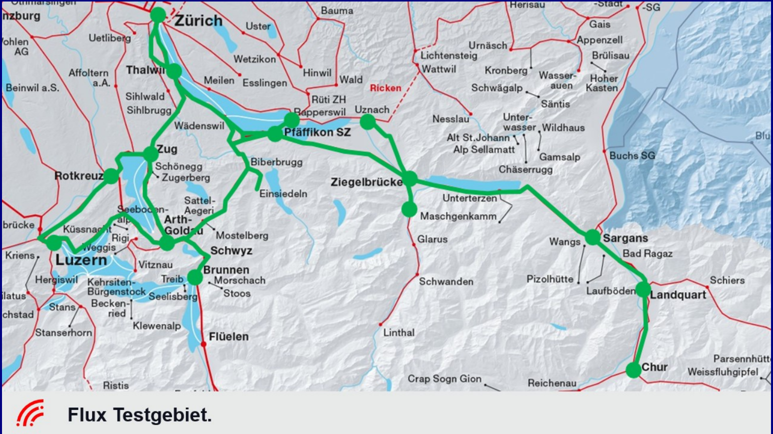
Motivation

- Increasing railway traffic demands global and flexible ways for scheduling trains in order to use railway networks to capacity
- Difficulty arises from dependencies among trains induced by connections and shared resources

- Train scheduling combines three distinct tasks
  - Routing
  - Conflict detection and resolution
  - Scheduling

- Solution operational at Swiss Federal Railway using *clingo*[DL]
  - ASP
  - Difference constraints
  - (Hybrid) Optimization
  - Heuristic directives
  - Multi-shot solving

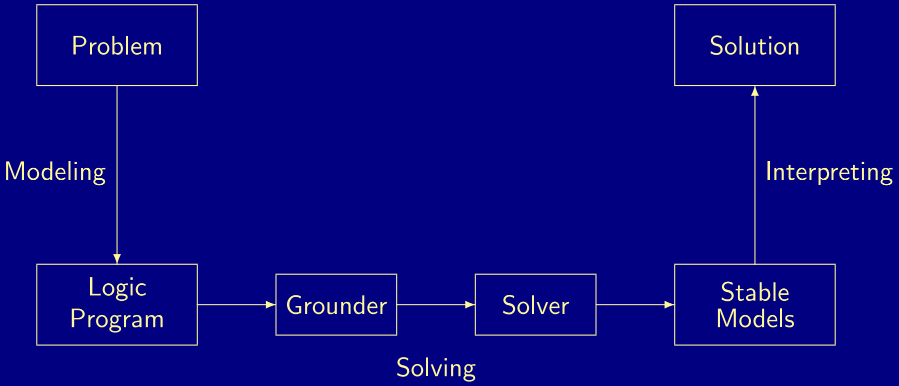
Benchmark



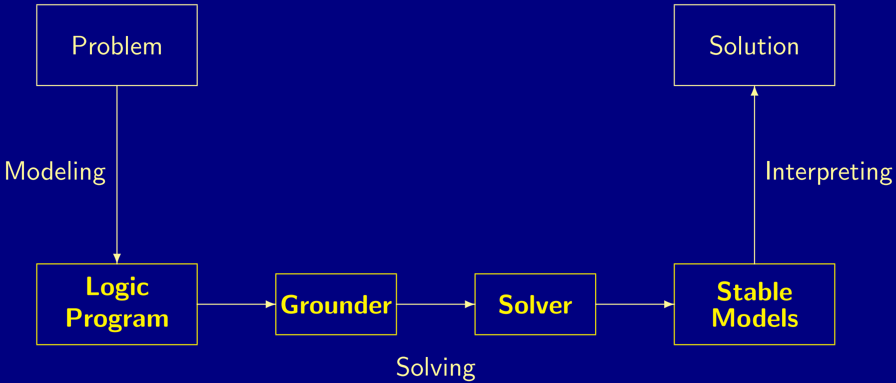
We optimally solved the train scheduling problem on real-world railway networks spanning about 150 km with up to 467 trains within 5 minutes

Potassco

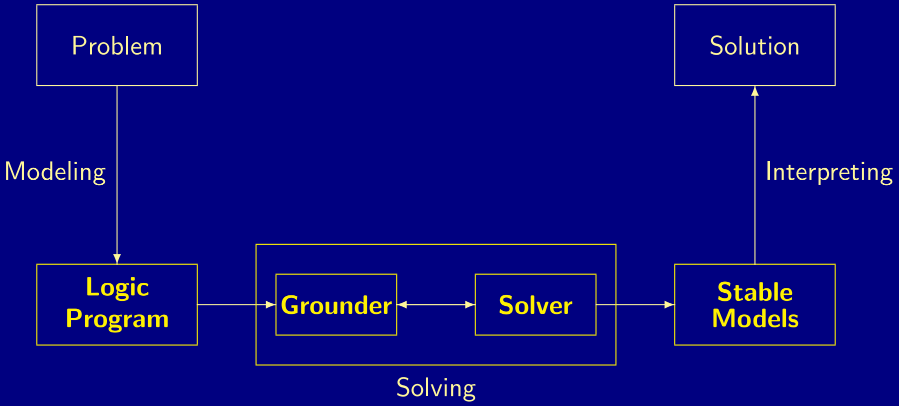
ASP solving process



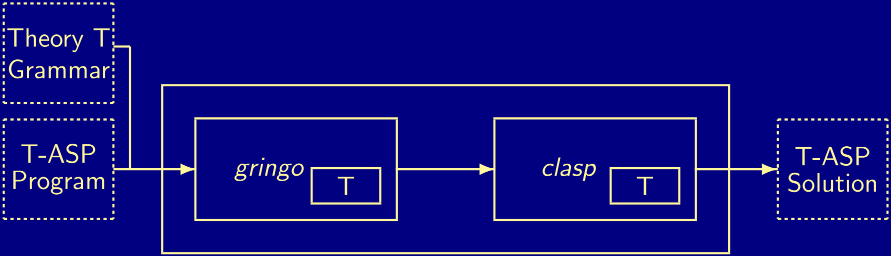
ASP solving process modulo theories



ASP solving process modulo theories



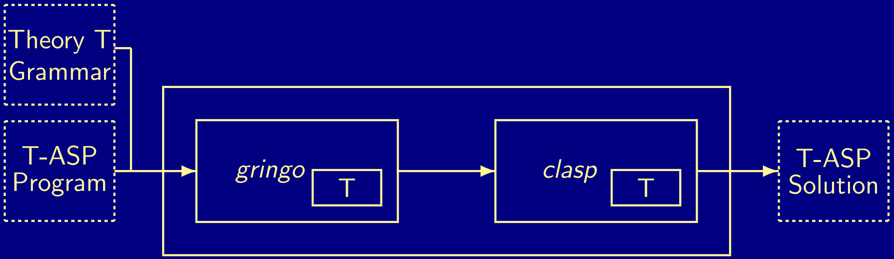
clingo's approach



- Challenge Logic programs with elusive theory atoms
- Example The atom "`&sum{x;-y}<=4`" stands for difference constraint $x - y \leq 4$

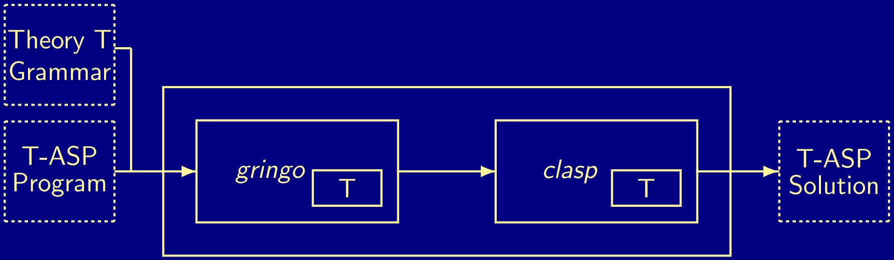
Potassco

clingo's approach



- Challenge Logic programs with elusive theory atoms
- Example The atom "&sum{x;-y}<=4" stands for difference constraint $x - y \leq 4$

clingo's approach



- Challenge Logic programs with elusive theory atoms
- Example The atom "`&sum{x;-y}<=4`" stands for difference constraint $x - y \leq 4$

Potassco

Open and Closed world reasoning

on numeric domains

- Closed world reasoning
  - if a variable occurs in true constraints, it is assigned appropriate values
  - if a variable occurs in no constraint, it is undefined

- Open world reasoning
  - if a variable occurs in true constraints, it is assigned appropriate values
  - if a variable occurs in no constraint, it takes all possible values

Potassco

Open and Closed world reasoning

on numeric domains

- Closed world reasoning
  - if a variable occurs in true constraints, it is assigned appropriate values
  - if a variable occurs in no constraint, it is undefined

- Open world reasoning
  - if a variable occurs in true constraints, it is assigned appropriate values
  - if a variable occurs in no constraint, it takes all possible values

Open and Closed world reasoning

on numeric domains

- Closed world reasoning
  - if a variable occurs in true constraints, it is assigned appropriate values
  - if a variable occurs in no constraint, it is undefined

  is non-monotonic

- Open world reasoning
  - if a variable occurs in true constraints, it is assigned appropriate values
  - if a variable occurs in no constraint, it takes all possible values

  is monotonic

Open and Closed world reasoning

on numeric domains

- Closed world reasoning
  - if a variable occurs in true constraints, it is assigned appropriate values
  - if a variable occurs in no constraint, it is undefined

  is non-monotonic

  offers defaults, succinctness
- Open world reasoning
  - if a variable occurs in true constraints, it is assigned appropriate values
  - if a variable occurs in no constraint, it takes all possible values

  is monotonic

HT_c Syntax

- Signature $\langle \mathcal{X}, \mathcal{D}, \mathcal{A} \rangle$
  - $\mathcal{X}$ variables
  - $\mathcal{D}$ domain
  - $\mathcal{A}$ atoms
- Note The syntax of atoms is left open
- Example Atom "$x - y \leq d$" with $x, y \in \mathcal{X}$ and $d \in \mathcal{D}$
- HT_c-formula φ over $\mathcal{A}$

$$\varphi \ ::= \ \bot \mid a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \quad \text{where } a \in \mathcal{A}$$

HT_c Syntax

- Signature $\langle \mathcal{X}, \mathcal{D}, \mathcal{A} \rangle$
  - $\mathcal{X}$ variables
  - $\mathcal{D}$ domain
  - $\mathcal{A}$ atoms
- Note The syntax of atoms is left open
- Example Atom "$x - y \leq d$" with $x, y \in \mathcal{X}$ and $d \in \mathcal{D}$
- HT_c-formula φ over $\mathcal{A}$

$$\varphi \ ::= \ \bot \mid a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \quad \text{where } a \in \mathcal{A}$$

Potassco

HT_c Syntax

- Signature $\langle \mathcal{X}, \mathcal{D}, \mathcal{A} \rangle$
  - $\mathcal{X}$ variables
  - $\mathcal{D}$ domain
  - $\mathcal{A}$ atoms
- Note The syntax of atoms is left open
- Example Atom "$x - y \leq d$" with $x, y \in \mathcal{X}$ and $d \in \mathcal{D}$
- HT_c-formula φ over $\mathcal{A}$

$$\varphi \ ::= \ \bot \mid a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \quad \text{where } a \in \mathcal{A}$$

Potassco

HT_c Semantics

- Valuation $v : \mathcal{X} \rightarrow \mathcal{D} \cup \{\boldsymbol{u}\}$
  - $\boldsymbol{u} \notin \mathcal{X} \cup \mathcal{D}$ stands for undefined

  Set-based representation $v \subseteq \mathcal{X} \times \mathcal{D}$
  - $(x, c) \in v$ and $(x, d) \in v$ implies $c = d$
  - $(x, d) \notin v$ if $v(x) = \boldsymbol{u}$

  $\mathcal{V}$ is the set of all valuations over $\mathcal{X}$ and $\mathcal{D}$
- Atom denotation $[\![\cdot]\!] : \mathcal{A} \rightarrow 2^{\mathcal{V}}$
- Example

$$[\![\text{“}x - y \leq d\text{”}]\!] = \{v \in \mathcal{V} \mid v(x), v(y),\ d \in \mathbb{Z},\ v(x) - v(y) \leq d\}$$

HT_c Semantics

- Valuation $v : \mathcal{X} \rightarrow \mathcal{D} \cup \{\boldsymbol{u}\}$
  - $\boldsymbol{u} \notin \mathcal{X} \cup \mathcal{D}$ stands for undefined

  Set-based representation $v \subseteq \mathcal{X} \times \mathcal{D}$
  - $(x, c) \in v$ and $(x, d) \in v$ implies $c = d$
  - $(x, d) \notin v$ if $v(x) = \boldsymbol{u}$

  $\mathcal{V}$ is the set of all valuations over $\mathcal{X}$ and $\mathcal{D}$

- Atom denotation $[\![\cdot]\!] : \mathcal{A} \rightarrow 2^{\mathcal{V}}$
- Example

$$[\![\text{“}x - y \leq d\text{”}]\!] = \{v \in \mathcal{V} \mid v(x), v(y),\ d \in \mathbb{Z},\ v(x) - v(y) \leq d\}$$

HT_c Semantics

- Valuation $v : \mathcal{X} \to \mathcal{D} \cup \{\boldsymbol{u}\}$
  - $\boldsymbol{u} \notin \mathcal{X} \cup \mathcal{D}$ stands for undefined

  Set-based representation $v \subseteq \mathcal{X} \times \mathcal{D}$
  - $(x, c) \in v$ and $(x, d) \in v$ implies $c = d$
  - $(x, d) \notin v$ if $v(x) = \boldsymbol{u}$

  $\mathcal{V}$ is the set of all valuations over $\mathcal{X}$ and $\mathcal{D}$
- Atom denotation $[\![\cdot]\!] : \mathcal{A} \to 2^{\mathcal{V}}$
- Example

$$[\![\text{“}x - y \leq d\text{”}]\!] = \{v \in \mathcal{V} \mid v(x), v(y),\ d \in \mathbb{Z},\ v(x) - v(y) \leq d\}$$

Potassco

HT_c-satisfaction

- HT_c-interpretation over $\mathcal{X},\mathcal{D}$ is a pair $\langle h, t\rangle$ of valuations over $\mathcal{X},\mathcal{D}$ such that $h \subseteq t$

- An HT_c-interpretation $\langle h, t\rangle$ satisfies a formula φ, written $\langle h, t\rangle \models \varphi$, if the following conditions hold
  1. $\langle h, t\rangle \not\models \bot$
  2. $\langle h, t\rangle \models a$ if both $h \in [\![a]\!]$ and $t \in [\![a]\!]$ for $a \in \mathcal{A}$
  3. $\langle h, t\rangle \models \varphi \wedge \psi$ if $\langle h, t\rangle \models \varphi$ and $\langle h, t\rangle \models \psi$
  4. $\langle h, t\rangle \models \varphi \vee \psi$ if $\langle h, t\rangle \models \varphi$ or $\langle h, t\rangle \models \psi$
  5. $\langle h, t\rangle \models \varphi \rightarrow \psi$ if $\langle h', t\rangle \not\models \varphi$ or $\langle h', t\rangle \models \psi$ for both $h' = h$ and $h' = t$.

HT_c-satisfaction

- HT_c-interpretation over $\mathcal{X}$,$\mathcal{D}$ is a pair $\langle h, t\rangle$ of valuations over $\mathcal{X}$,$\mathcal{D}$ such that $h \subseteq t$

- An HT_c-interpretation $\langle h, t\rangle$ satisfies a formula φ, written $\langle h, t\rangle \models \varphi$, if the following conditions hold
  1. $\langle h, t\rangle \not\models \bot$
  2. $\langle h, t\rangle \models a$ if both $h \in [\![\, a \,]\!]$ and $t \in [\![\, a \,]\!]$ for $a \in \mathcal{A}$
  3. $\langle h, t\rangle \models \varphi \wedge \psi$ if $\langle h, t\rangle \models \varphi$ and $\langle h, t\rangle \models \psi$
  4. $\langle h, t\rangle \models \varphi \vee \psi$ if $\langle h, t\rangle \models \varphi$ or $\langle h, t\rangle \models \psi$
  5. $\langle h, t\rangle \models \varphi \rightarrow \psi$ if $\langle h', t\rangle \not\models \varphi$ or $\langle h', t\rangle \models \psi$ for both $h' = h$ and $h' = t$.

HT_c-equilibrium model

- A total interpretation $\langle t, t\rangle$ is an equilibrium model of a formula φ, if
  1. $\langle t, t\rangle \models \varphi$
  2. $\langle h, t\rangle \not\models \varphi$ for all $h \subset t$
- t is called an HT_c-stable model of φ

Potassco

HT_c-equilibrium model

- A total interpretation $\langle t, t \rangle$ is an equilibrium model of a formula φ, if
  1. $\langle t, t \rangle \models \varphi$
  2. $\langle h, t \rangle \not\models \varphi$ for all $h \subset t$
- t is called an HT_c-stable model of φ

Potassco

HT_c benefits

- Semantic framework for capturing ASP modulo theory systems combining closed and open world reasoning
  - conservative extension of HT
  - flexibility due to open syntax and denotational semantics
  - study of AMT systems
  - study of language fragments
  - soundness of program transformations
  - warrant substitution of equivalent expressions
  - etc.

Potassco

Outline

More features of interest

- Meta programming
- Qualitative and quantitative optimization
- Heuristic programming
- Application interface programming
  - Multi-shot solving
  - Theory solving
- Linear Temporal and Dynamic reasoning
- Visualization

- Playful? `https://potassco.org`

Potassco

More features of interest

- Meta programming
- Qualitative and quantitative optimization
- Heuristic programming
- Application interface programming
  - Multi-shot solving
  - Theory solving
- Linear Temporal and Dynamic reasoning
- Visualization

- Playful? `https://potassco.org`

Potassco

Outline

Take home message

Take home message

Modeling + Grounding + Solving

Take home message

Modeling + Grounding + Solving

ASP = DB+LP+KR+SAT

Take home message

Modeling + Grounding + Solving

ASP = DB+LP+KR+SMT^n

Take home message

Modeling + Grounding + Solving

$\mathbf{ASP} = \mathbf{DB}+\mathbf{LP}+\mathbf{KR}+\mathbf{SMT}^{n}$

https://potassco.org

Potassco

Take home message

Modeling + Grounding + Solving

$$\textbf{ASP} = \textbf{DB}+\textbf{LP}+\textbf{KR}+\textbf{SMT}^n$$

https://potassco.org

And it's fun !