Knowledge-driven Artificial Intelligence

Torsten Schaub

University of Potsdam & Potassco Solutions GmbH

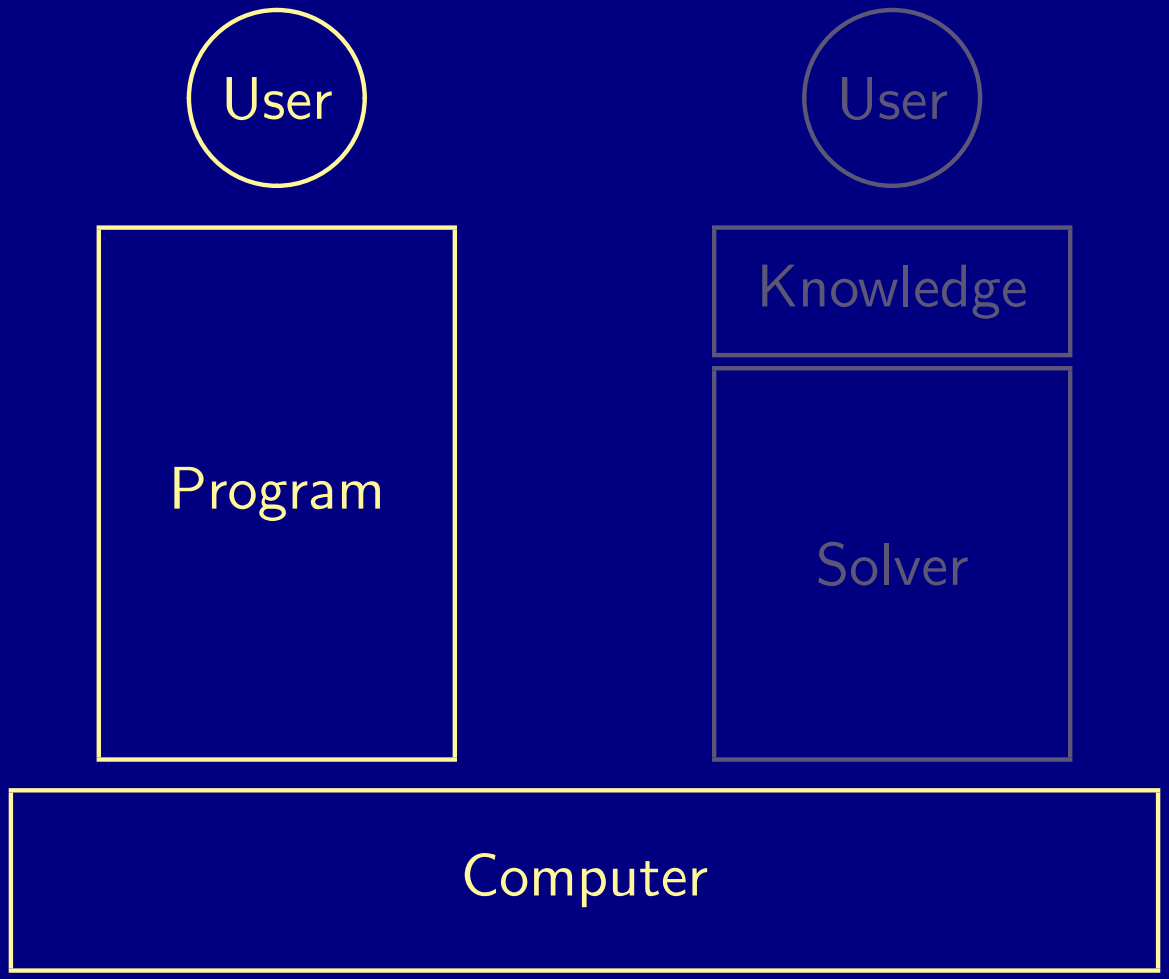
Outline

Potassco

Outline

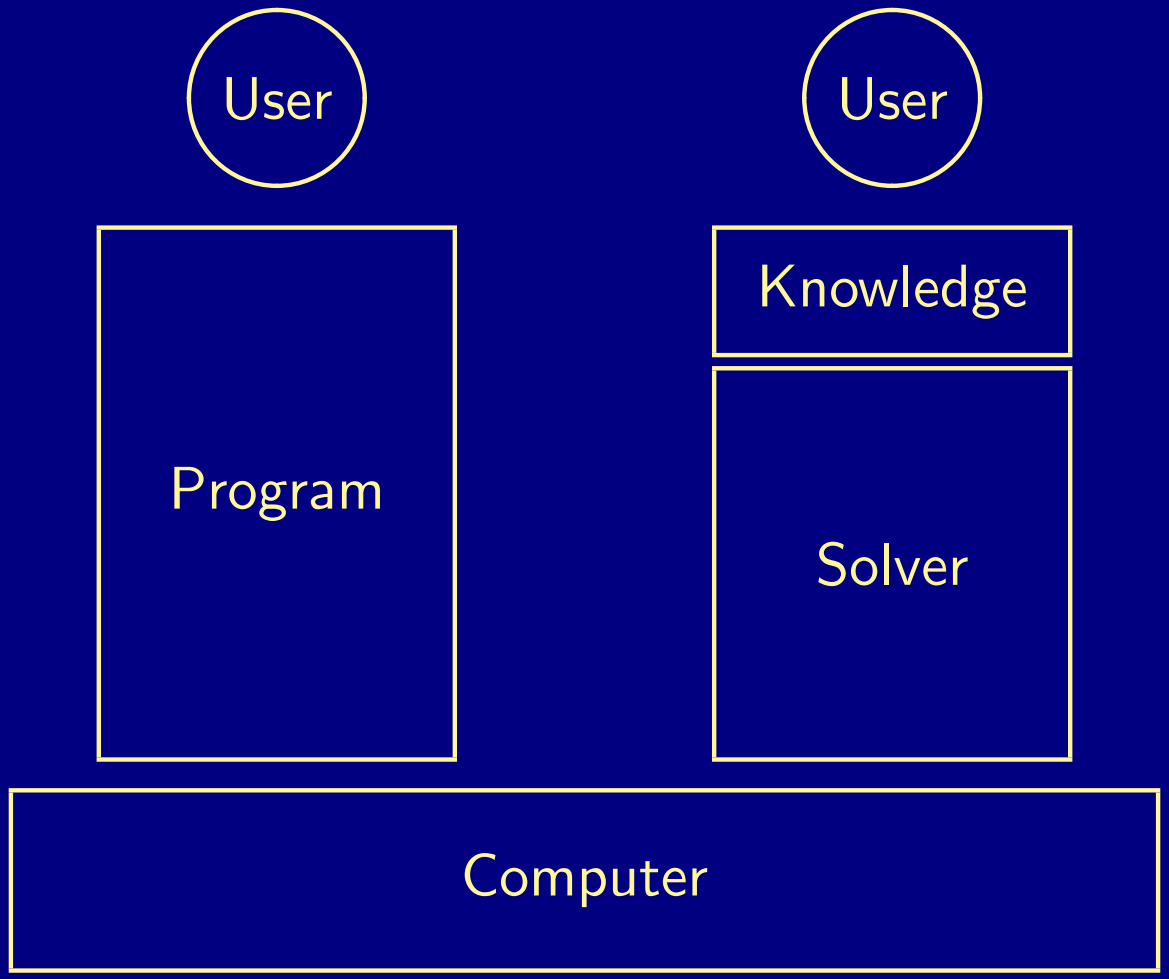
Potassco

Traditional Software



Potassco

Knowledge-driven Software

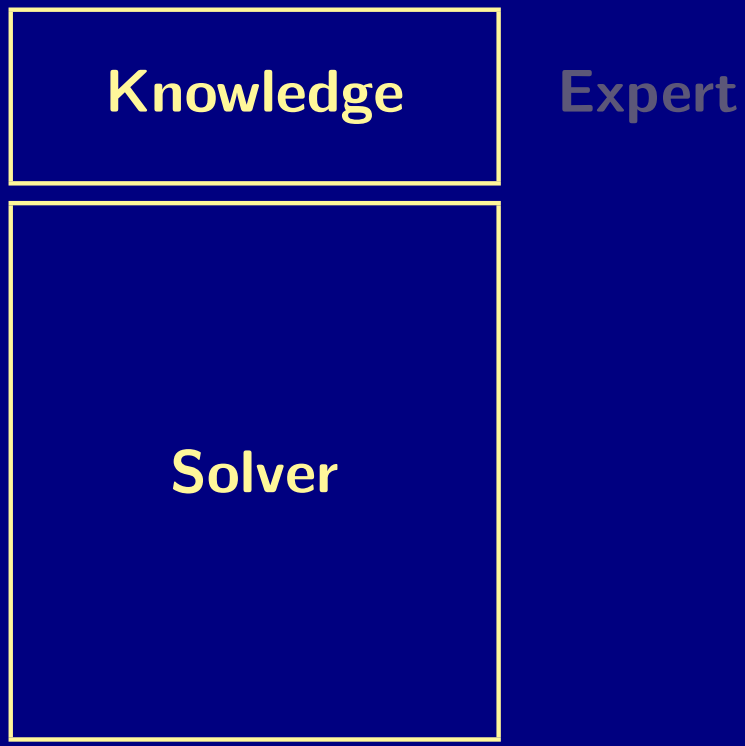


Potassco

What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability

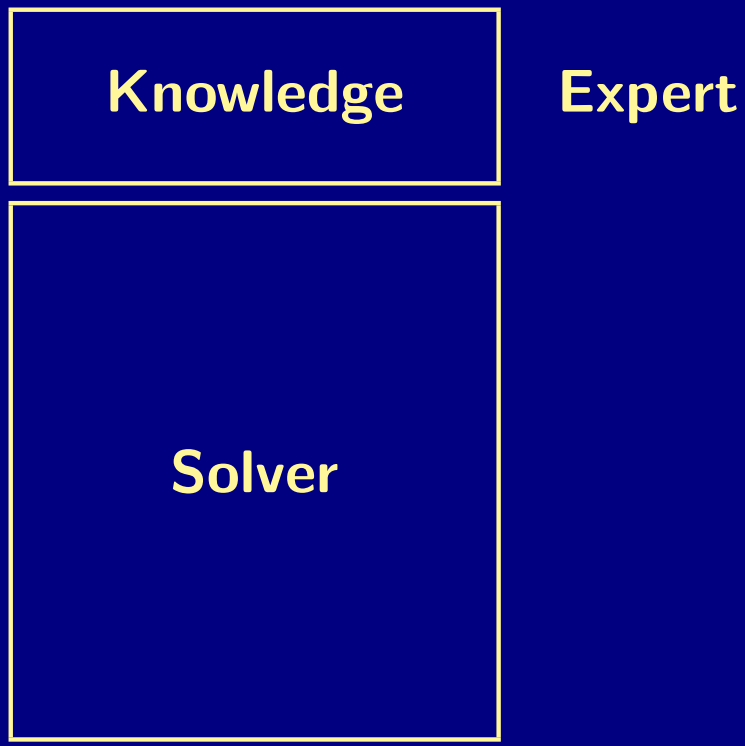


Potassco

What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability

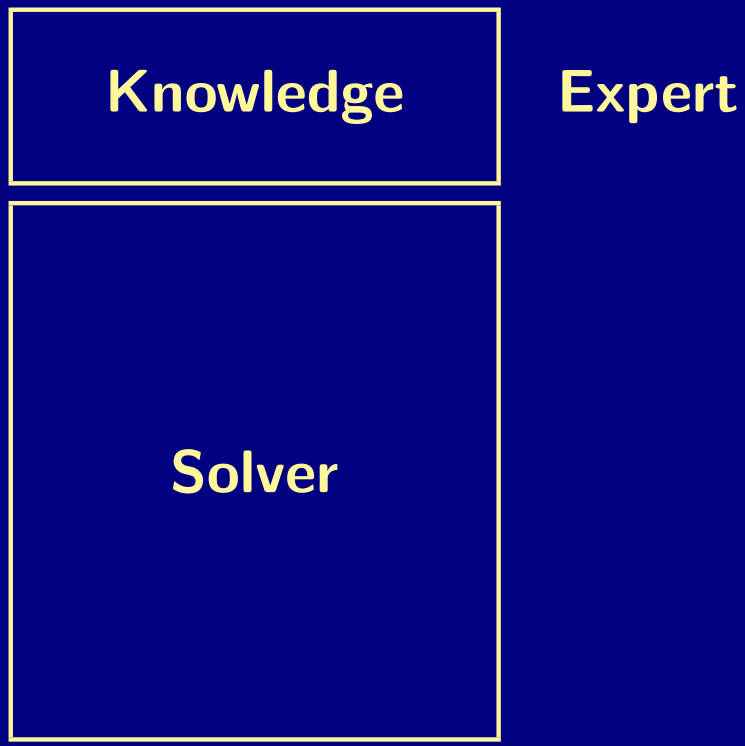


Potassco

What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability



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 (ASP)

- What is ASP?
  ASP is an approach for declarative problem solving

Potassco

Answer Set Programming (ASP)

- What is ASP?
  ASP is an approach for declarative problem solving
- Where is ASP from?
  - Databases
  - Logic programming
  - Knowledge representation and reasoning
  - Satisfiability solving

Potassco

Answer Set Programming (ASP)

- What is ASP? **ASP = DB+LP+KR+SAT!**
  ASP is an approach for declarative problem solving
- Where is ASP from?
  - Databases
  - Logic programming
  - Knowledge representation and reasoning
  - Satisfiability solving

Potassco

Answer Set Programming (ASP)

- What is ASP?
  ASP is an approach for declarative problem solving
- What is ASP good for?
  Solving knowledge-intense combinatorial (optimization) problems

Potassco

Answer Set Programming (ASP)

- What is ASP?
  ASP is an approach for declarative problem solving
- What is ASP good for?
  Solving knowledge-intense combinatorial (optimization) problems
- What problems are this?
  Problems consisting of (many) decisions and constraints

Potassco

Answer Set Programming (ASP)

- What is ASP?
  ASP is an approach for declarative problem solving
- What is ASP good for?
  Solving knowledge-intense combinatorial (optimization) problems
- What problems are this?
  Problems consisting of (many) decisions and constraints

  Examples Sudoku, Configuration, Diagnosis, Music composition, Planning, System design, Time tabling, etc.

Potassco

Answer Set Programming (ASP)

- What is ASP?
  ASP is an approach for declarative problem solving
- What is ASP good for?
  Solving knowledge-intense combinatorial (optimization) problems
- What problems are this? — **And industrial ones ?**
  Problems consisting of (many) decisions and constraints

  Examples Sudoku, Configuration, Diagnosis, Music composition, Planning, System design, Time tabling, etc.

Potassco

Answer Set Programming (ASP)

- What is ASP?
  ASP is an approach for declarative problem solving
- What is ASP good for?
  Solving knowledge-intense combinatorial (optimization) problems
- What problems are this? — **And industrial ones ?**
  - Debian, Ubuntu: Linux package configuration
  - Exeura: Call routing
  - FCC: Radio frequency auction
  - Gioia Tauro: Workforce management
  - NASA: Decision support for Space Shuttle
  - SBB: Train disposition
  - Siemens: Partner units configuration
  - Variantum: Product configuration

Potassco

Answer Set Programming (ASP)

- What is ASP?
  ASP is an approach for declarative problem solving
- What is ASP good for?
  Solving knowledge-intense combinatorial (optimization) problems
- What problems are this? — **And industrial ones ?**
  - Debian, Ubuntu: Linux package configuration
  - Exeura: Call routing
  - **FCC: Radio frequency auction**
  - Gioia Tauro: Workforce management
  - NASA: Decision support for Space Shuttle
  - SBB: Train disposition
  - Siemens: Partner units configuration
  - Variantum: Product configuration

Potassco

Answer Set Programming (ASP)

- What is ASP?
  ASP is an approach for declarative pr
- What is ASP good for?
  Solving knowledge-intense combinator
- What problems are this? — **And indu**
  - Debian, Ubuntu: Linux package con
  - Exeura: Call routing
  - **FCC: Radio frequency auction**
  - Gioia Tauro: Workforce management
  - NASA: Decision support for Space Shuttle
  - SBB: Train disposition
  - Siemens: Partner units configuration
  - Variantum: Product configuration

Over 13 months in 2016–17 the US Federal Communications Commission conducted an "incentive auction" to repurpose radio spectrum from broadcast television to wireless internet. In the end, the auction yielded $19.8 billion, $10.05 billion of which was paid to 175 broadcasters for voluntarily relinquishing their licenses across 14 UHF channels. Stations that continued broadcasting were assigned potentially new channels to fit as densely as possible into the channels that remained. The government netted more than $7 billion (used to pay down the national debt) after covering costs. A crucial element of the auction design was the construction of a solver, dubbed SATFC, that determined whether sets of stations could be "repacked" in this way; it needed to run every time a station was given a price quote. This

Potassco

Answer Set Programming (ASP)

- What is ASP?
  ASP is an approach for declarative problem solving
- What is ASP good for?
  Solving knowledge-intense combinatorial (optimization) problems
- What problems are this?
  Problems consisting of (many) decisions and constraints
- What are ASP's distinguishing features?
  - High level, versatile modeling language
  - High performance solvers
  - Qualitative and quantitative optimization

Potassco

Answer Set Programming (ASP)

- What is ASP?
  ASP is an approach for declarative problem solving
- What is ASP good for?
  Solving knowledge-intense combinatorial (optimization) problems
- What problems are this?
  Problems consisting of (many) decisions and constraints
- What are ASP's distinguishing features?
  - High level, versatile modeling language
  - High performance solvers
  - Qualitative and quantitative optimization
- Any industrial impact?
  - ASP Tech companies: DLV Systems and **Potassco Solutions**
  - Increasing interest in (large) companies

Potassco

Outline

Potassco

Some (biased) moments in time

- '80 Capturing incomplete information
- '90 Amalgamation and computation
- '00 Applications and semantic rediscoveries
- '10 Customization and integration

Potassco

Some (biased) moments in time

- '80 Capturing incomplete information
  - Databases Closed world assumption
  - Logic programming Negation as failure
  - Non-monotonic reasoning
    Auto-epistemic and Default logics, Circumscription
- '90 Amalgamation and computation
- '00 Applications and semantic rediscoveries
- '10 Customization and integration

Potassco

Some (biased) moments in time

- '80 Capturing incomplete information
- '90 Amalgamation and computation
  - Logic programming semantics
    Well-founded and stable models semantics
  - ASP solving
    "Stable models = Well-founded semantics + Branch"
- '00 Applications and semantic rediscoveries
- '10 Customization and integration

Potassco

Some (biased) moments in time

- '80 Capturing incomplete information
- '90 Amalgamation and computation
- '00 Applications and semantic rediscoveries
  - Growing dissemination — see last slides —
  - Constructive logics Equilibrium Logic
- '10 Customization and integration

Potassco

Some (biased) moments in time

- '80 Capturing incomplete information
- '90 Amalgamation and computation
- '00 Applications and semantic rediscoveries
- '10 Customization and integration
  - Complex reasoning modes APIs, multi-shot solving
  - Hybridization Constraint ASP, theory solving

Potassco

Some (biased) moments in time

- '80 Capturing incomplete information
- '90 Amalgamation and computation
- '00 Applications and semantic rediscoveries
- '10 Customization and integration

Potassco

Some (biased) moments in time

- '80 Capturing incomplete information
- '90 Amalgamation and computation
- '00 Applications and semantic rediscoveries
- '10 Customization and integration
- '20 Real-world industries

Potassco

Some (biased) moments in time

- '80 Capturing incomplete information
- '90 Amalgamation and computation
- '00 Applications and semantic rediscoveries
- '10 Customization and integration
- '20 Real-world industries
  - Industrial applications
  - Software engineering

Potassco

Outline

Potassco

Logic programs

- A logic program, P, over a set $\mathcal{A}$ of atoms is a finite set of rules
- A rule is of the form

$$\mathtt{a}_0 \texttt{ :- } \mathtt{a}_1, \ldots, \mathtt{a}_m, \texttt{ not } \mathtt{a}_{m+1}, \ldots, \texttt{ not } \mathtt{a}_n.$$

  where $0 \leq m \leq n$ and each $a_i \in \mathcal{A}$ is an atom for $0 \leq i \leq n$

Potassco

Logic programs

- A logic program, P, over a set $\mathcal{A}$ of atoms is a finite set of rules
- A rule is of the form

$$\underbrace{\mathtt{a}_0}_{\text{head}} \;\mathtt{:-}\; \underbrace{\mathtt{a}_1,\ldots,\mathtt{a}_m,\; \mathtt{not}\; \mathtt{a}_{m+1},\ldots,\; \mathtt{not}\; \mathtt{a}_n.}_{\text{body}}$$

where $0 \leq m \leq n$ and each $a_i \in \mathcal{A}$ is an atom for $0 \leq i \leq n$

Potassco

Logic programs

- A logic program, P, over a set $\mathcal{A}$ of atoms is a finite set of rules
- A rule is of the form

$$\mathtt{a}_0 \texttt{ :- } \mathtt{a}_1, \ldots, \mathtt{a}_m, \texttt{ not } \mathtt{a}_{m+1}, \ldots, \texttt{ not } \mathtt{a}_n.$$

  where $0 \leq m \leq n$ and each $a_i \in \mathcal{A}$ is an atom for $0 \leq i \leq n$

Potassco

Logic programs

- A logic program, P, over a set $\mathcal{A}$ of atoms is a finite set of rules
- A rule is of the form

  $$\mathtt{a}_0 \texttt{ :- } \mathtt{a}_1, \ldots, \mathtt{a}_m, \texttt{ not } \mathtt{a}_{m+1}, \ldots, \texttt{ not } \mathtt{a}_n.$$

  where $0 \leq m \leq n$ and each $a_i \in \mathcal{A}$ is an atom for $0 \leq i \leq n$

- Semantics given by stable models, informally,
  models of P justifying each true atom by a proof

Potassco

Logic programs

- A logic program, P, over a set $\mathcal{A}$ of atoms is a finite set of rules
- A rule is of the form

$$\mathtt{a}_0 \texttt{ :- } \mathtt{a}_1, \ldots, \mathtt{a}_m, \texttt{ not } \mathtt{a}_{m+1}, \ldots, \texttt{ not } \mathtt{a}_n.$$

  where $0 \leq m \leq n$ and each $a_i \in \mathcal{A}$ is an atom for $0 \leq i \leq n$

- Semantics given by stable models, informally,
  models of P justifying each true atom by a proof

  Minimal models in the logic HT (Heyting'30) / G3 (Gödel'32)

Potassco

Open and Closed world reasoning

- 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

- 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

Potassco

Open and Closed world reasoning

- 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

- 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

Potassco

Open and Closed world reasoning

- 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
- 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 and Closed world reasoning

- 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
- 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, reachability, succinctness

Potassco

Open and Closed world reasoning

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

Potassco

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 $\{a\}$

Potassco

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 $\varphi ::= \bot \mid a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi$
- 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 $\varphi ::= \bot \mid a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi$
- 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 $\varphi ::= \bot \mid a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi$
- 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 $\varphi ::= \bot \mid a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi$
- 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 $\varphi ::= \bot \mid a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi$
- 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

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 $\langle H, T\rangle \models \varphi$

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 $\langle H, T\rangle \models \varphi$

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 $\langle H, T\rangle \models \varphi$

Potassco

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**

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

(Pearce'96)

- 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
  - $\langle H, T\rangle \models P$ iff $H \models P^T$ (P^T is the reduct of P by T)

Potassco

Equilibrium models

(Pearce'96)

- 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
  - $\langle H, T\rangle \models P$ iff $H \models P^T$ (P^T is the reduct of P by T)

Potassco

Equilibrium models

(Pearce'96)

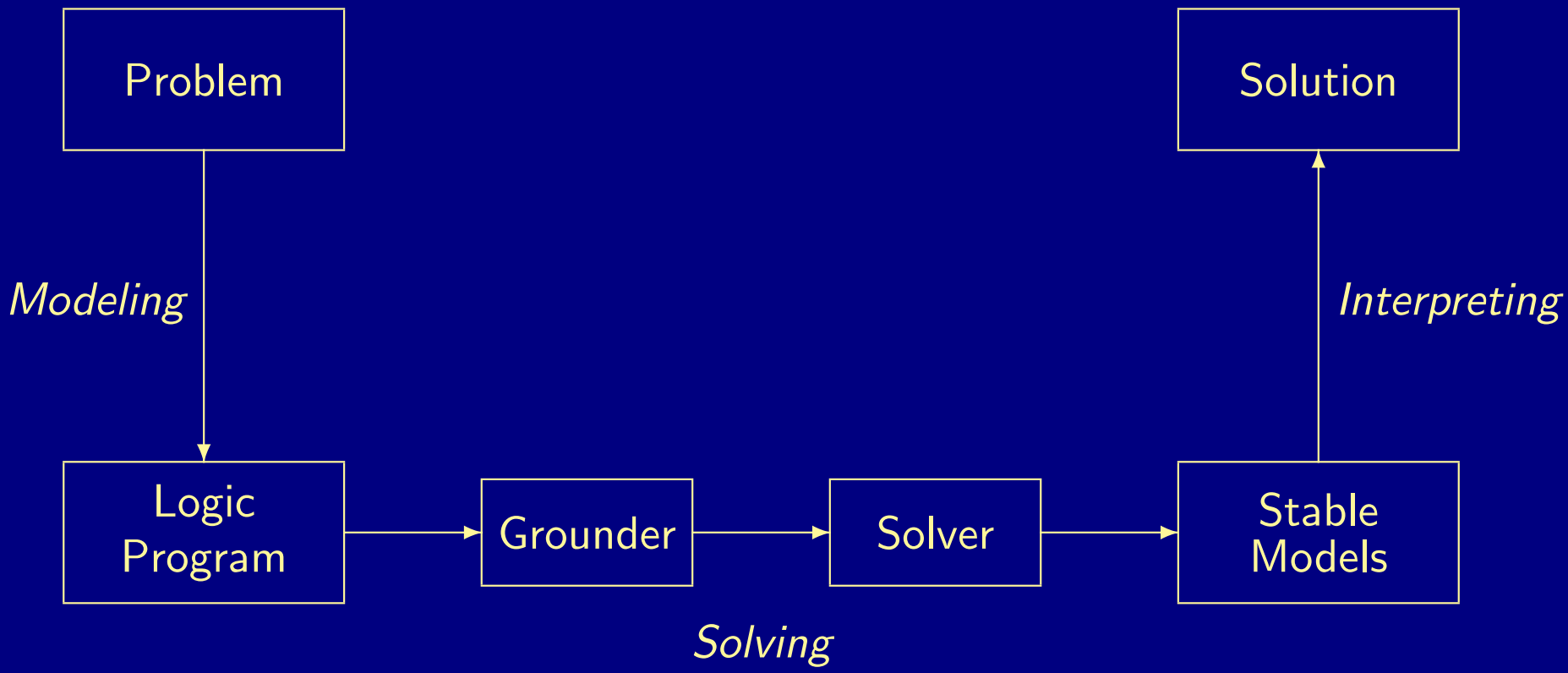
- A total interpretation $\langle T, T\rangle$ is an equilibrium model of a formula φ, if
  1. $\langle T, T\rangle \models \varphi$
  2. $\langle H, T\rangle \not\models \varphi$ for all $H \subset T$
- T is called a stable model of φ
- Note
  - $\langle T, T\rangle$ acts as a classical model
  - $\langle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

Potassco

Outline

Potassco

Modeling, grounding, and solving



Potassco

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).
```

Potassco

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) }.
```

Potassco

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]
```

Potassco

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
```

Potassco

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
```

Potassco

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
```

Potassco

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
```

Potassco

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
```

Potassco

Some of our industrial projects

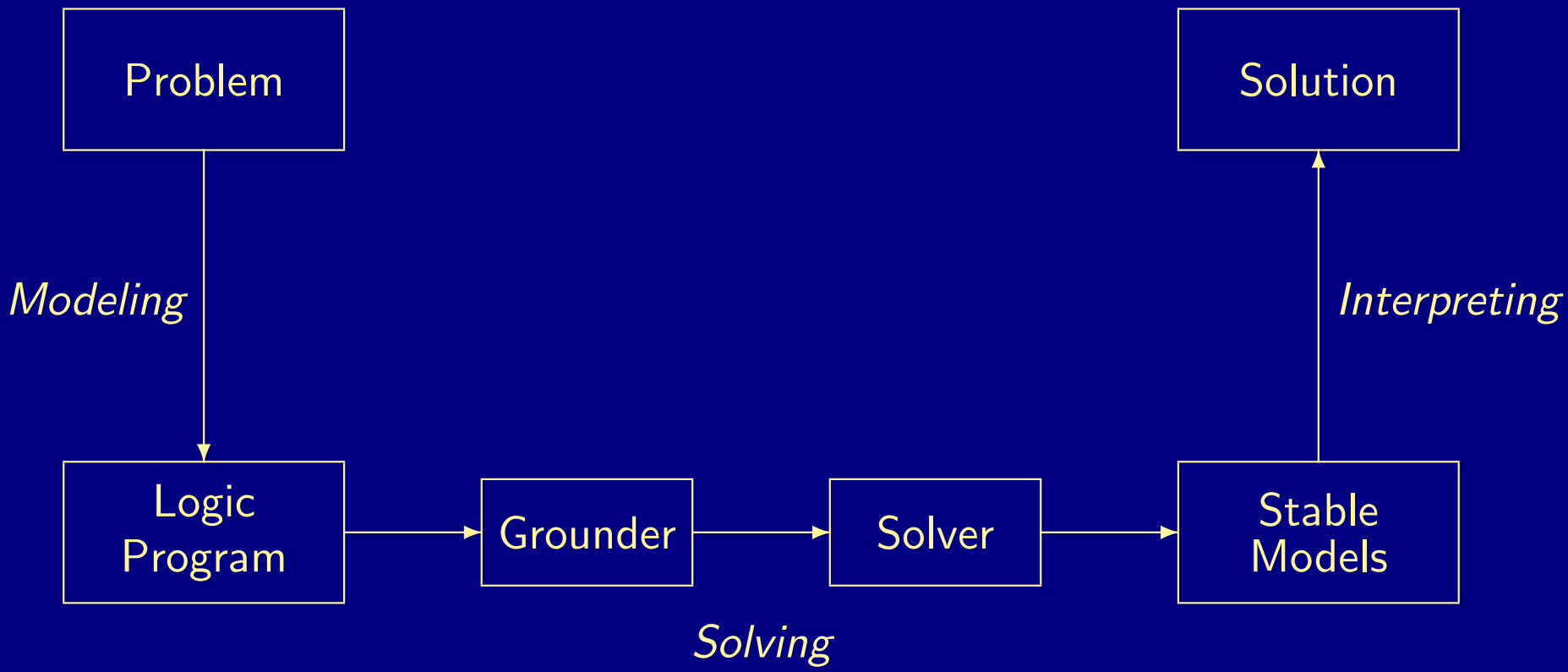
- Railway companies
  - Shift planning
  - Room planning
  - Maintenance scheduling
  - Train disposition
- Automobile industries
  - Car assembly sequencing
  - Car version configuration
  - Robotic vehicle control
- Logistics industries
  - Delivery optimization
  - Stock optimization
- Manufacturing industries
  - Product configuration
  - Machine configuration
  - Assembly line layout
- Public services
  - University timetabling
  - University study regulations

Potassco

Outline

Potassco

Modeling, grounding, and solving



Potassco

clingraph

Sudoku

Model

ASP Developer

A problem

"A sudoku puzzle"

Visualization encoding

Potassco

clingraph

Sudoku

Model

```
initial(1,4,2). initial(3,3,1).
initial(3,4,3). initial(3,1,4).
```

ASP Developer

Visualization encoding

Potassco

clingraph

Sudoku[1]

Model

```
initial(1,4,2). initial(3,3,1).
initial(3,4,3). initial(3,1,4).
```

ASP Developer


```
  sudoku(X,Y,V) :- initial(X,Y,V).
{ sudoku(X,Y,V) : val(V) } = 1 :- pos(X,Y).
:- sudoku(X,Y,V), sudoku(X',Y,V), X != X'.
:- sudoku(X,Y,V), sudoku(X,Y',V), Y != Y'.
:- sudoku(X,Y,V), sudoku(X',Y',V), (X,Y)!=(X',Y'),
  subgrid(X,Y,S), subgrid(X',Y',S).
```

Visualization encoding

[1] Omitting pos/2, val/1, subgrid/2

Potassco

clingraph

Sudoku

Model

ASP Developer

```
initial(1,4,2). initial(3,3,1).
initial(3,4,3). initial(3,1,4).
```

```
  sudoku(X,Y,V) :- initial(X,Y,V).
{ sudoku(X,Y,V) : val(V) } = 1 :- pos(X,Y).
:- sudoku(X,Y,V), sudoku(X',Y,V), X != X'.
:- sudoku(X,Y,V), sudoku(X,Y',V), Y != Y'.
:- sudoku(X,Y,V), sudoku(X',Y',V), (X,Y)!=(X',Y'),
  subgrid(X,Y,S), subgrid(X',Y',S).
```

```
clingo version 5.7.1
Reading from sudoku.lp ..
Solving...
Answer: 1
sudoku(1,4,2) sudoku(3,3,1)
sudoku(1,2,3) sudoku(3,4,3)
sudoku(3,1,4) sudoku(1,1,1)
sudoku(4,2,1) sudoku(3,2,2)
sudoku(4,1,3) sudoku(1,3,4)
sudoku(4,3,2) sudoku(4,4,4)
sudoku(2,1,2) sudoku(2,4,1)
sudoku(2,3,3) sudoku(2,2,4)
SATISFIABLE

Models : 1+
Calls : 1
Time : 0.007s
CPU Time : 0.003s
```

Visualization encoding

Potassco

clingraph

Sudoku

Model

```
initial(1,4,2). initial(3,3,1).
initial(3,4,3). initial(3,1,4).
```

ASP Developer

```
   sudoku(X,Y,V) :- initial(X,Y,V).
 { sudoku(X,Y,V) : val(V) } = 1 :- pos(X,Y).
:- sudoku(X,Y,V), sudoku(X',Y,V), X != X'.
:- sudoku(X,Y,V), sudoku(X,Y',V), Y != Y'.
:- sudoku(X,Y,V), sudoku(X',Y',V), (X,Y)!=(X',Y'),
   subgrid(X,Y,S), subgrid(X',Y',S).
```

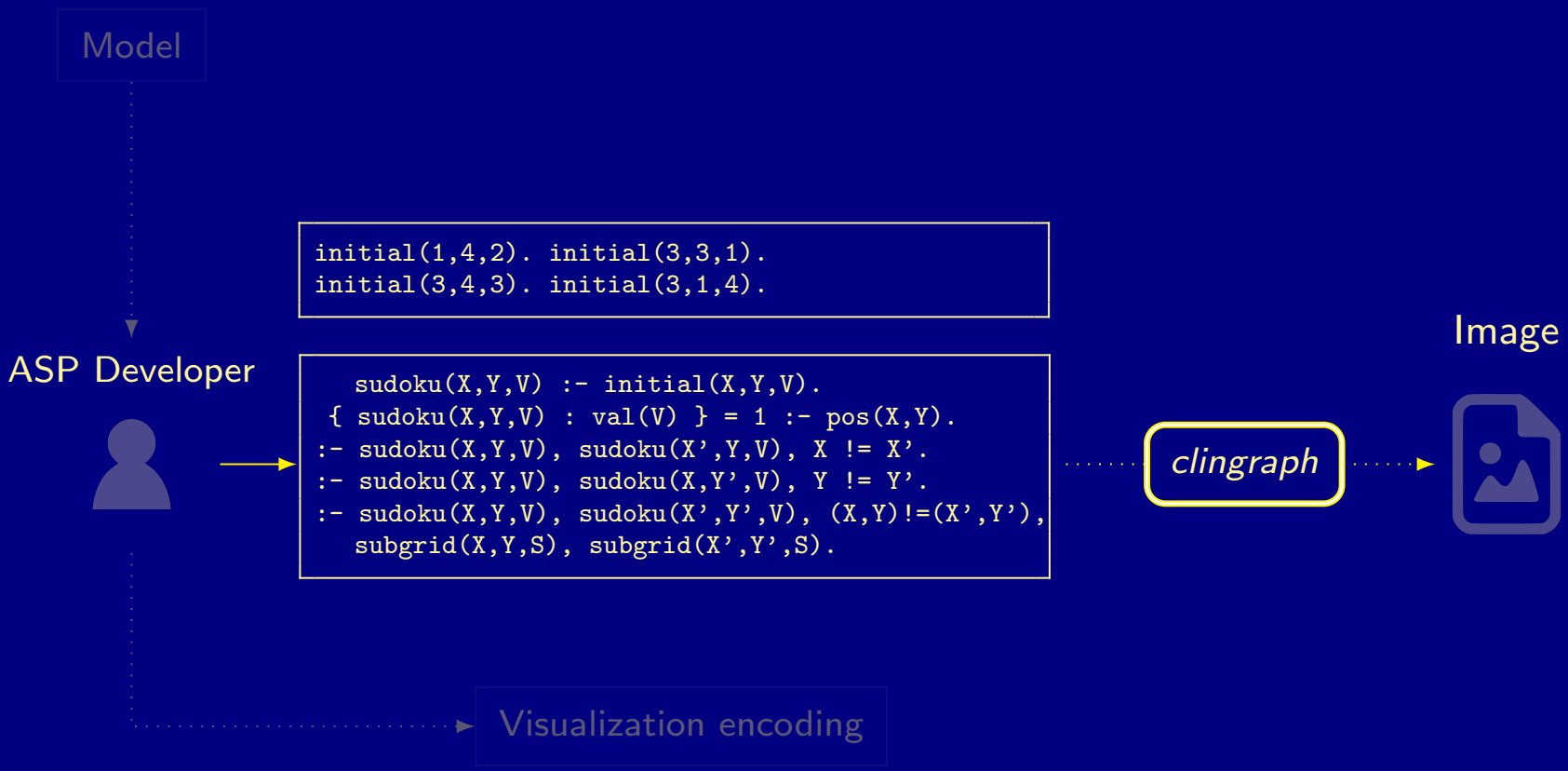
Image

Visualization encoding

Potassco

clingraph

Sudoku



Potassco

clingraph

Sudoku

Model

```
initial(1,4,2). initial(3,3,1).
initial(3,4,3). initial(3,1,4).
```

ASP Developer

```
  sudoku(X,Y,V) :- initial(X,Y,V).
{ sudoku(X,Y,V) : val(V) } = 1 :- pos(X,Y).
:- sudoku(X,Y,V), sudoku(X',Y,V), X != X'.
:- sudoku(X,Y,V), sudoku(X,Y',V), Y != Y'.
:- sudoku(X,Y,V), sudoku(X',Y',V), (X,Y)!=(X',Y'),
  subgrid(X,Y,S), subgrid(X',Y',S).
```

Image

Visualization encoding

Potassco

clingraph

Sudoku

Model

```
initial(1,4,2). initial(3,3,1).
initial(3,4,3). initial(3,1,4).
```

ASP Developer

```
   sudoku(X,Y,V) :- initial(X,Y,V).
 { sudoku(X,Y,V) : val(V) } = 1 :- pos(X,Y).
:- sudoku(X,Y,V), sudoku(X',Y,V), X != X'.
:- sudoku(X,Y,V), sudoku(X,Y',V), Y != Y'.
:- sudoku(X,Y,V), sudoku(X',Y',V), (X,Y)!=(X',Y'),
   subgrid(X,Y,S), subgrid(X',Y',S).
```

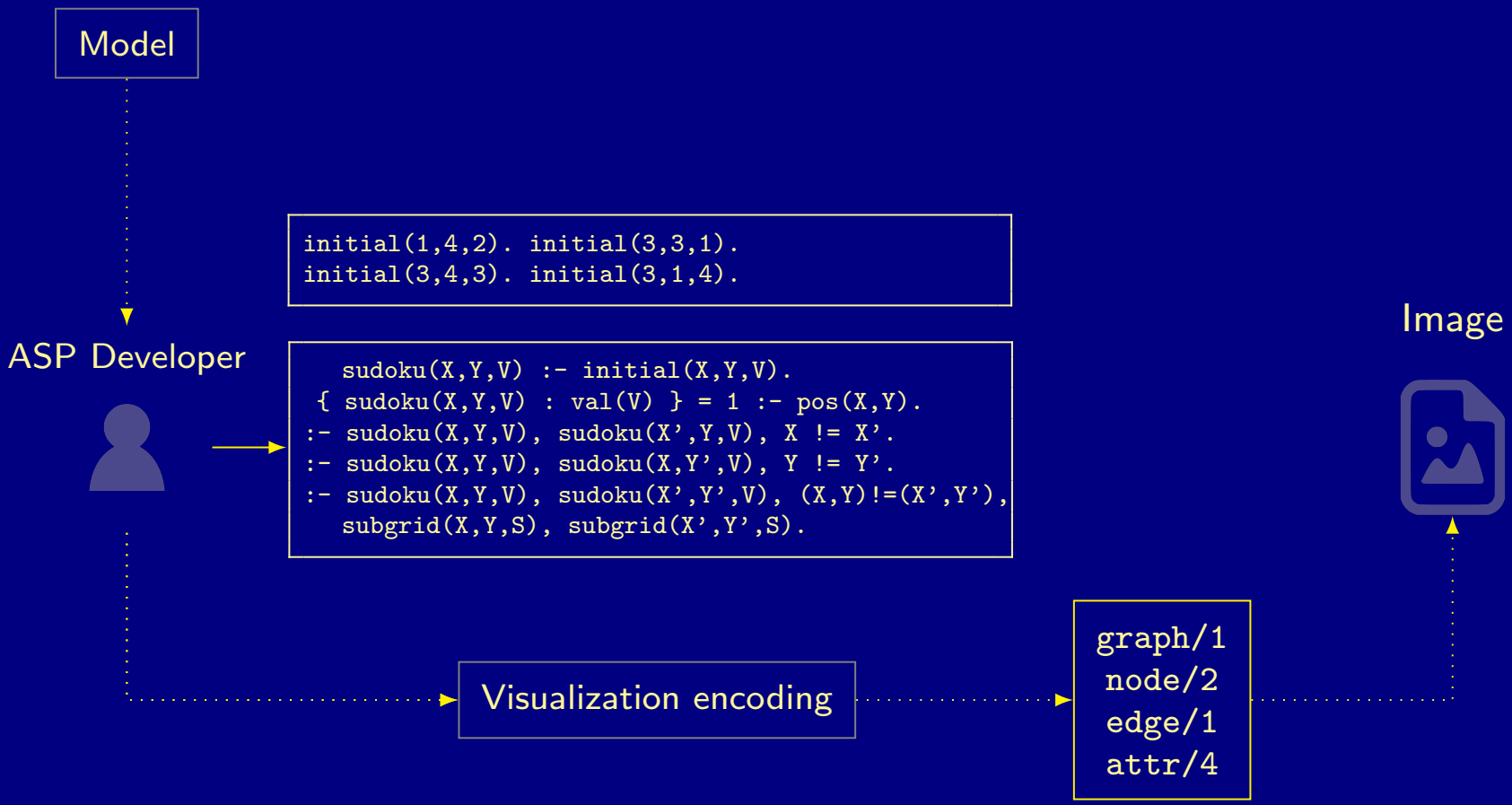
Image

Visualization encoding

Potassco

clingraph

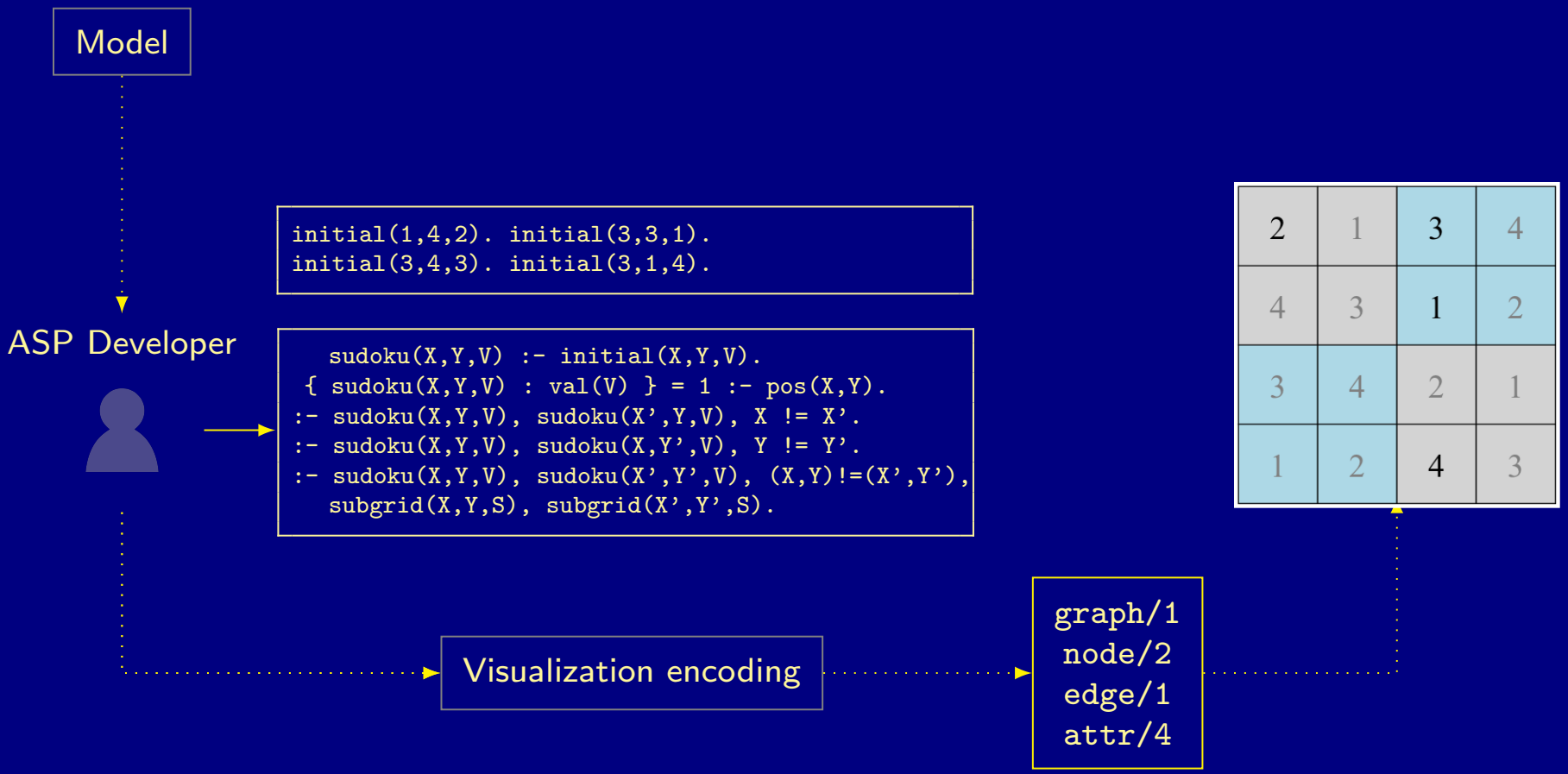
Sudoku



Potassco

clingraph

Sudoku



Potassco

Visualization encoding

Sudoku

2	1	3	4
4	3	1	2
3	4	2	1
1	2	4	3

- `graph/1`
- `node/2`
- `edge/1`
- `attr/4`

```
graph(sudoku).

node(pos(X,Y), sudoku) :- pos(X,Y).

attr(node, pos(X,Y), label, V) :- sudoku(X,Y,V).
attr(node, pos(X,Y), shape, square) :- sudoku(X,Y,V).
attr(node, pos(X,Y), style, filled) :- sudoku(X,Y,V).
attr(node, pos(X,Y), fontsize, 30) :- sudoku(X,Y,V).
attr(node, pos(X,Y), width, "1") :- sudoku(X,Y,V).
attr(node, pos(X,Y), pos, @pos(X,Y)) :- pos(X,Y).
attr(node, pos(X,Y), fillcolor, blue) :- pos(X,Y), (((X-1)/dim)+((Y-1)/dim))\2==0.
attr(node, pos(X,Y), fontcolor, gray50) :- sudoku(X,Y,V), not initial(X,Y,_).
```

Potassco

Visualization encoding

Sudoku

2	1	3	4
4	3	1	2
3	4	2	1
1	2	4	3

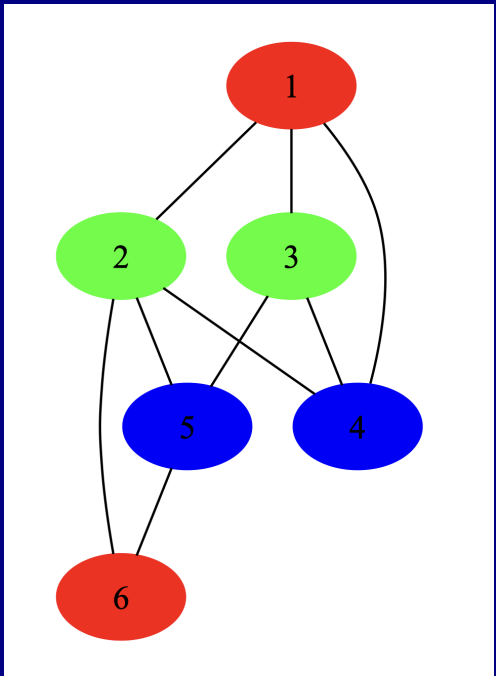
- `graph/1`
- `node/2`
- `edge/1`
- `attr/4`

```
graph(sudoku).

node(pos(X,Y), sudoku) :- pos(X,Y).

attr(node, pos(X,Y), label, V) :- sudoku(X,Y,V).
attr(node, pos(X,Y), shape, square) :- sudoku(X,Y,V).
attr(node, pos(X,Y), style, filled) :- sudoku(X,Y,V).
attr(node, pos(X,Y), fontsize, 30) :- sudoku(X,Y,V).
attr(node, pos(X,Y), width, "1") :- sudoku(X,Y,V).
attr(node, pos(X,Y), pos, @pos(X,Y)) :- pos(X,Y).
attr(node, pos(X,Y), fillcolor, blue) :- pos(X,Y), (((X-1)/dim)+((Y-1)/dim))\2==0.
attr(node, pos(X,Y), fontcolor, gray50) :- sudoku(X,Y,V), not initial(X,Y,_).
```

Potassco

Example

Simple graph

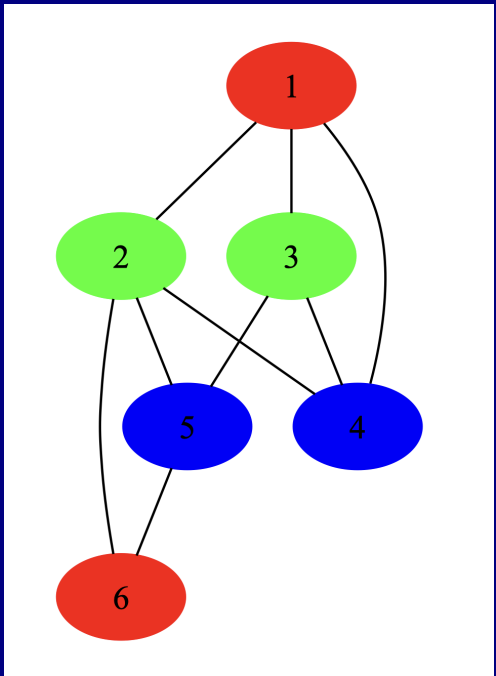



```
#show node/1.
#show edge((N,M)) : edge(N, M).
#show attr(node, N, style, filled): node(N).
#show attr(node, N, color, C) : assign(N, C).
```

Potassco

Example

Simple graph

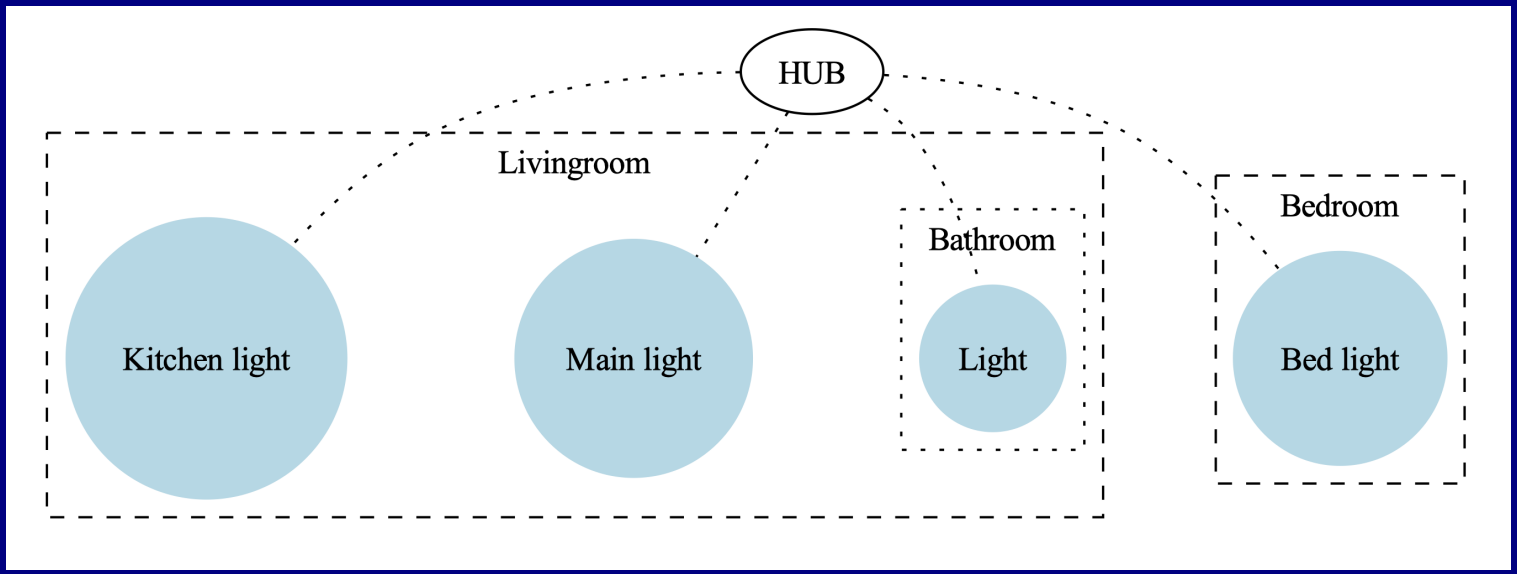



```
#show node/1.
#show edge((N,M)) : edge(N, M).
#show attr(node, N, style, filled): node(N).
#show attr(node, N, color, C) : assign(N, C).
```

Potassco

Example

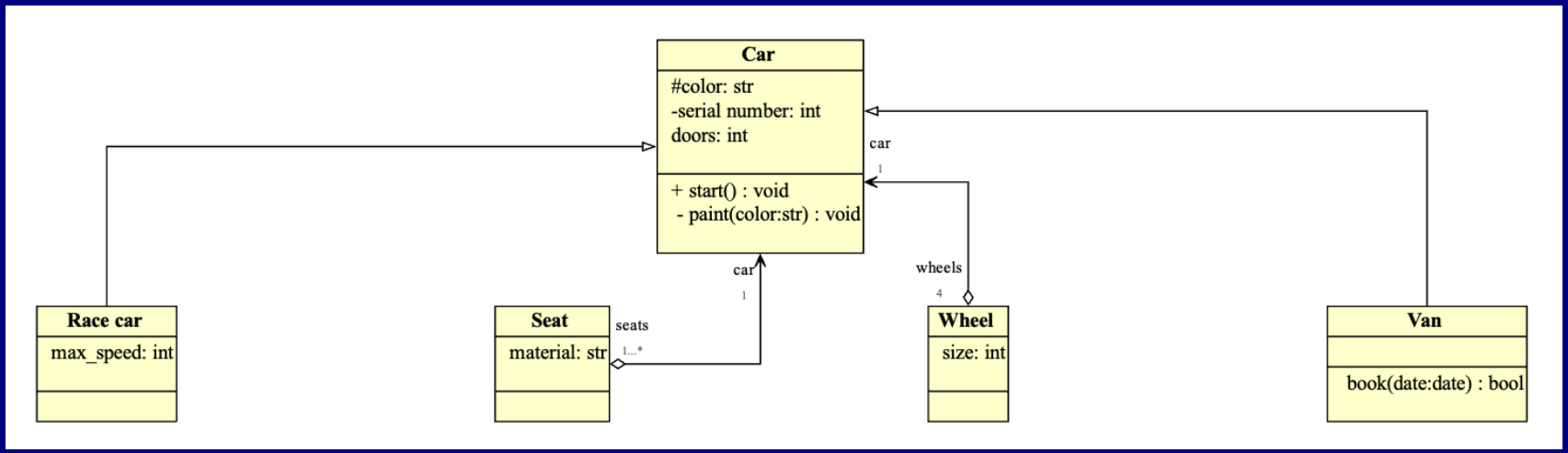
Subgraphs



Potassco

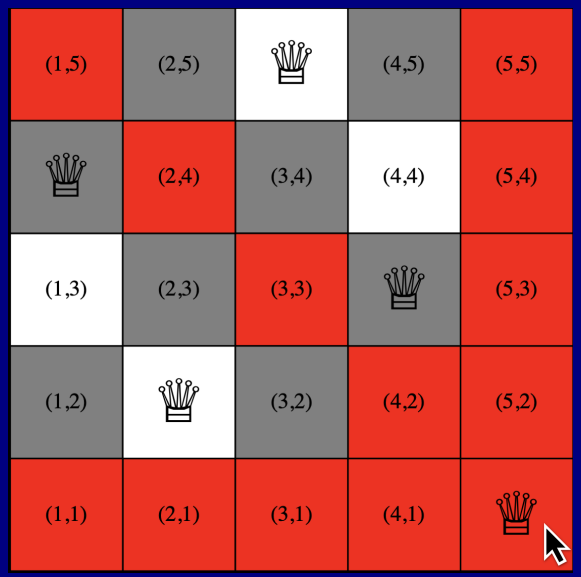
Example

UML



Potassco

Example
SVG



Potassco

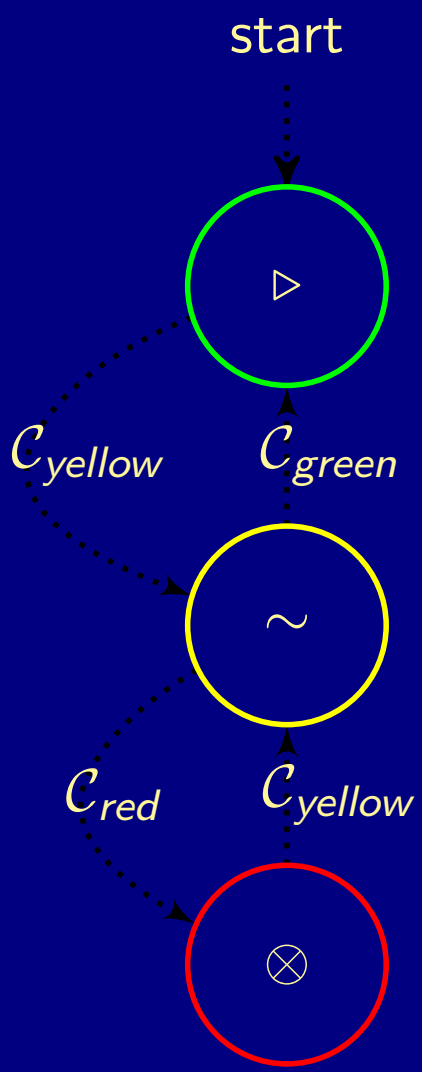
Example

SVG

Potassco

Example

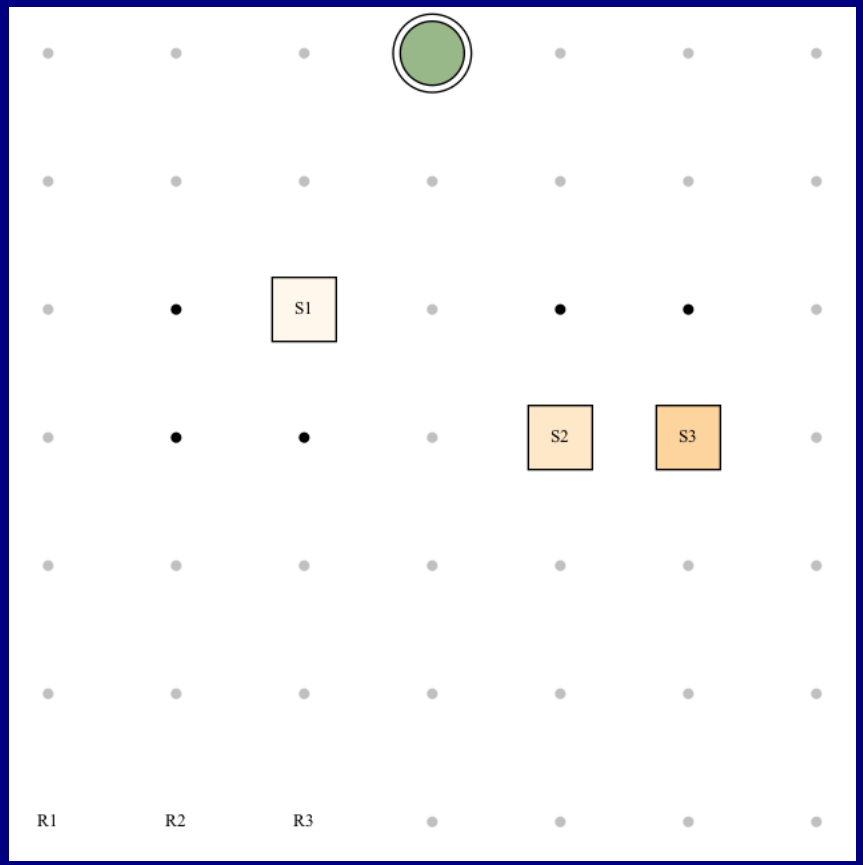
LaTeX



Potassco

Example

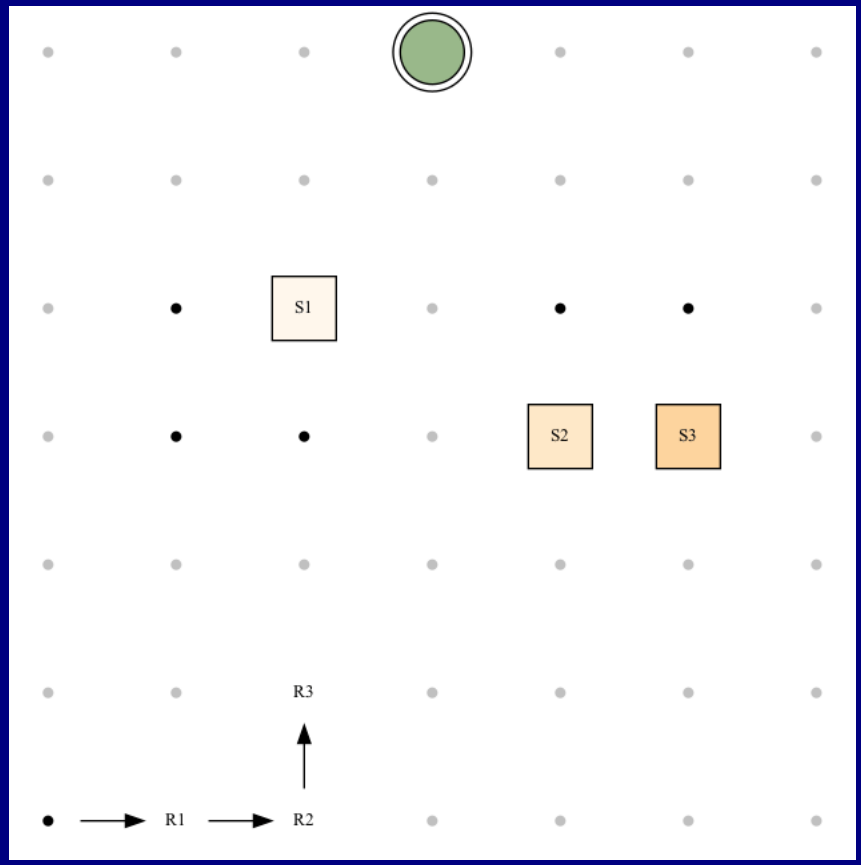
GIF animation



Potassco

Example

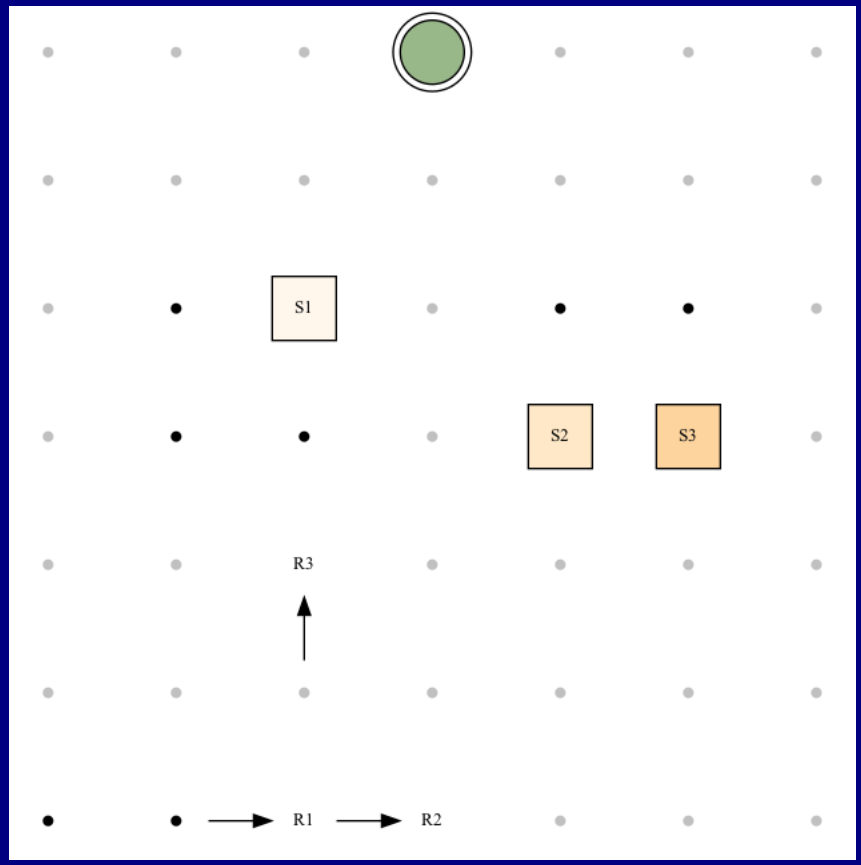
GIF animation



Potassco

Example

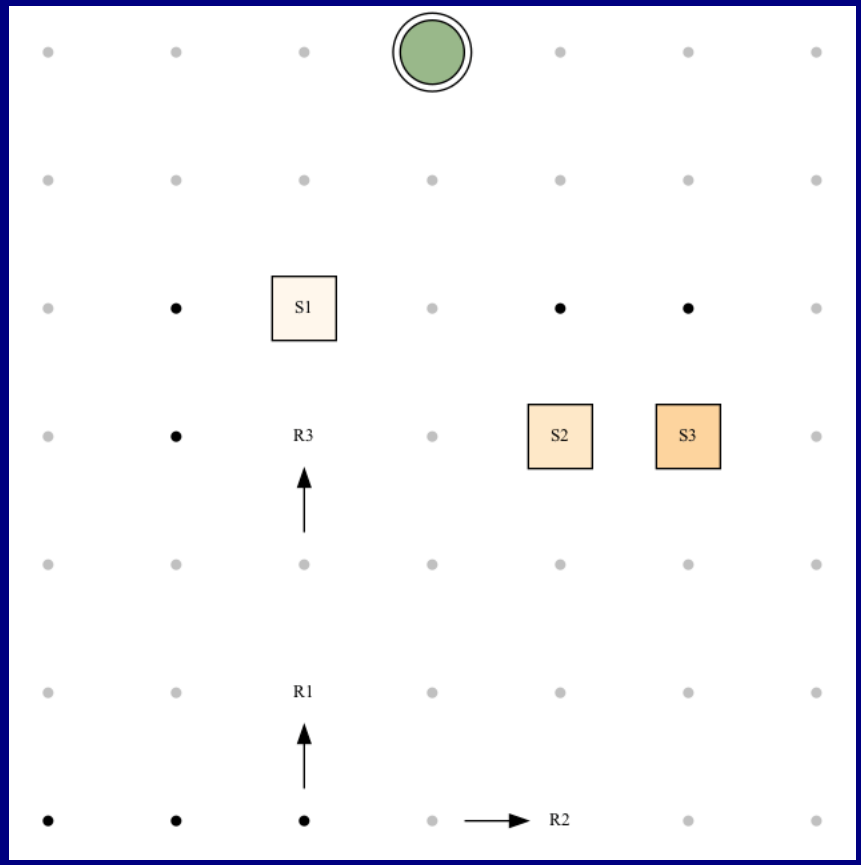
GIF animation



Potassco

Example

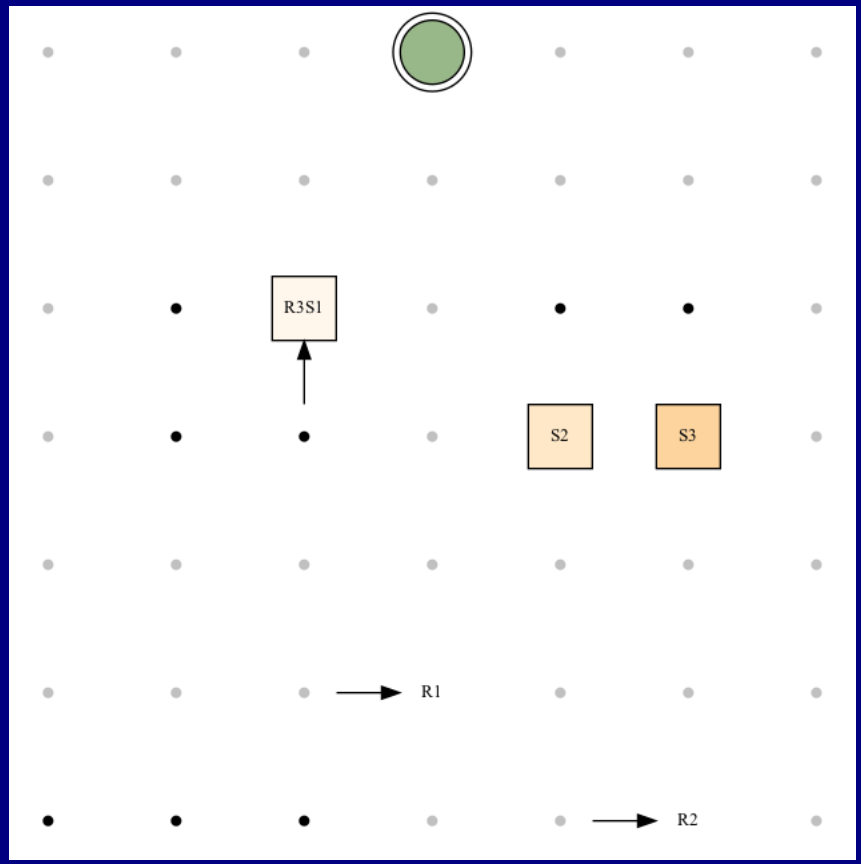
GIF animation



Potassco

Example

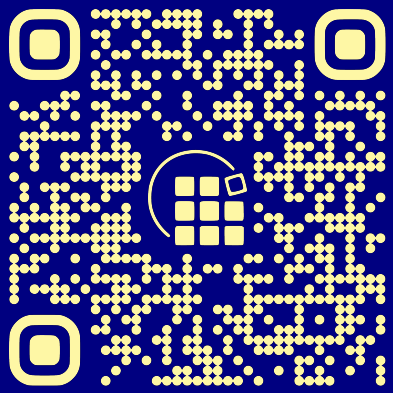
GIF animation



Potassco

clingraph

- An ASP-based front-end to *graphviz*
  - Define visualizations in terms of ASP
  - Visualize instances, solutions, or even the solving process
  - Extends ASP's rapid prototyping with visualization
- Functionalities
  - Command line usage
  - Python API
  - Different output formats
  - Integration with *clingo*
- Open source software
  - `https://github.com/potassco/clingraph`
  - `https://clingraph.readthedocs.io`



Potassco

Outline

Potassco

More features of interest

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

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

Potassco

Outline

Potassco

ASP systems and tools

gringo	grounding	*clinsight*	editing
clasp	solving	*xclingo*	explaining
clingo	ASP	*clingo-server*	surfing
metasp	Meta ASP	*acclingo*	tuning
clingcon	ASP+CP	*anthem*	verifying
fclingo	ASP+CP	*asprin*	preferring
clingo[DL]	ASP+CP	*clingraph*	visualizing
clingo[LP]	ASP+CP	*clinguin*	interacting
clingo[LPX]	ASP+CP	*viasp*	visualizing
eclingo	epistemic ASP	*clintest*	testing
plingo	probabilistic ASP	*clorm*	dataing
telingo	temporal ASP	*ngo*	optimizing

Potassco

ASP systems and tools

- *gringo* grounding
- *clasp* solving
- *clingo* ASP
- *metasp* Meta ASP
- *clingcon* ASP+CP
- *fclingo* ASP+CP
- *clingo*[DL] ASP+CP
- *clingo*[LP] ASP+CP
- *clingo*[LPX] ASP+CP
- *eclingo* epistemic ASP
- *plingo* probabilistic ASP
- *telingo* temporal ASP

- *clinsight* editing
- *xclingo* explaining
- *clingo-server* surfing
- *acclingo* tuning
- *anthem* verifying
- *asprin* preferring
- *clingraph* visualizing
- *clinguin* interacting
- *viasp* visualizing
- *clintest* testing
- *clorm* dataing
- *ngo* optimizing

Potassco

Application-oriented systems

aspartame	constraint solver
aspcafe	vehicle equipment specification
aspcud	software package configuration
asprilo	warehouse simulation
chasp	music composition
flatzingo	constraint solver
fluto	metabolic network expansion
plasp	planning system
qasp	quantified ASP solver
spa	study planner
teaspoon	university timetabling system
xorro	sampling stable models

Potassco

Outline

Potassco

Take home message

Potassco

Take home message

Modeling + Grounding + Solving

Potassco

Take home message

Modeling + Grounding + Solving

ASP = DB+LP+KR+SAT

Potassco

Take home message

Modeling + Grounding + Solving

ASP = DB+LP+KR+SMT^n

Potassco

Take home message

Modeling + Grounding + Solving

ASP = DB+LP+KR+SMT^n

https://potassco.org

Potassco

Take home message

Modeling + Grounding + Solving

ASP = DB+LP+KR+SMT^n

https://potassco.org

And it's fun !

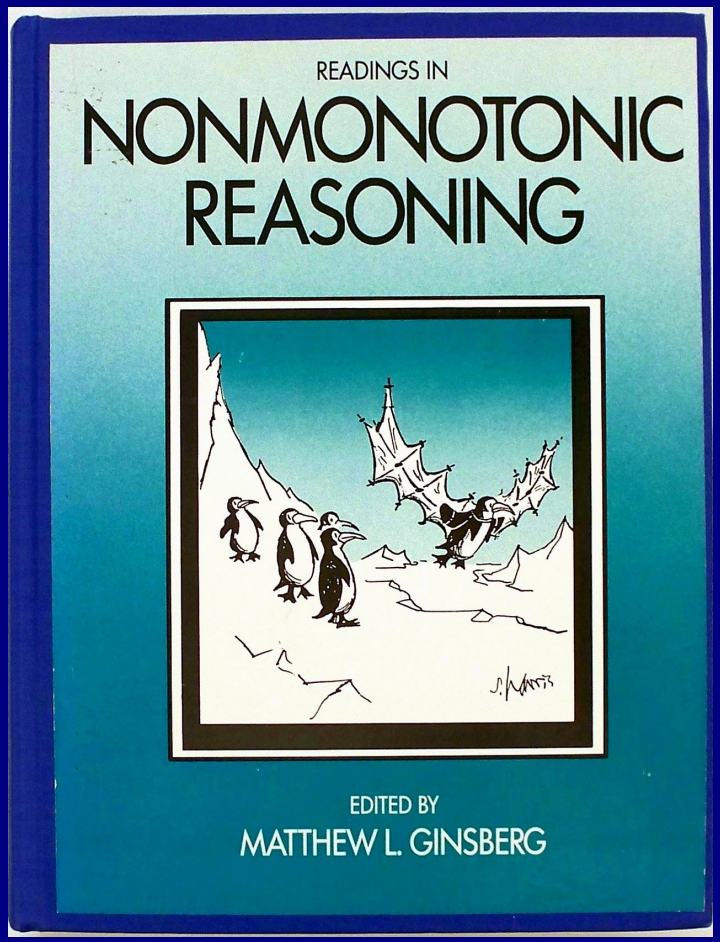
Potassco

Epilogue

After all, it's all about Tweety!

Epilogue

After all, it's all about Tweety!

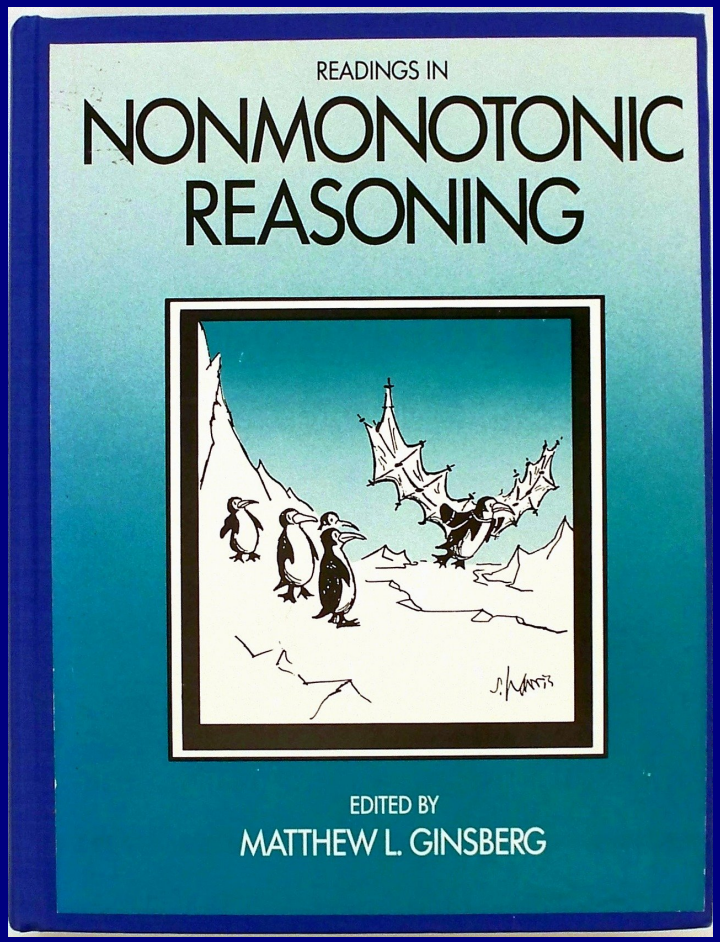


'87

Potassco

Epilogue

After all, it's all about Tweety!



'87

'25

Potassco