Answer Set Solving in Practice

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

Potassco Slide Packages are licensed under a Creative Commons Attribution 3.0 Unported License.

Motivation: Overview

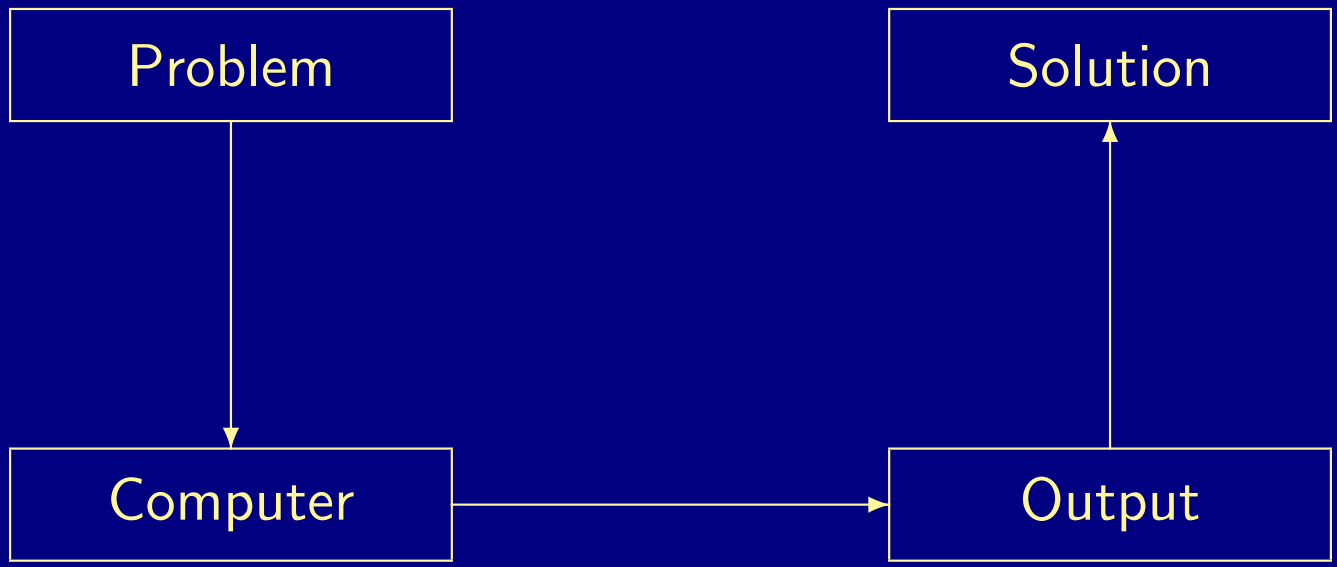
Potassco

Outline

Potassco

Informatics

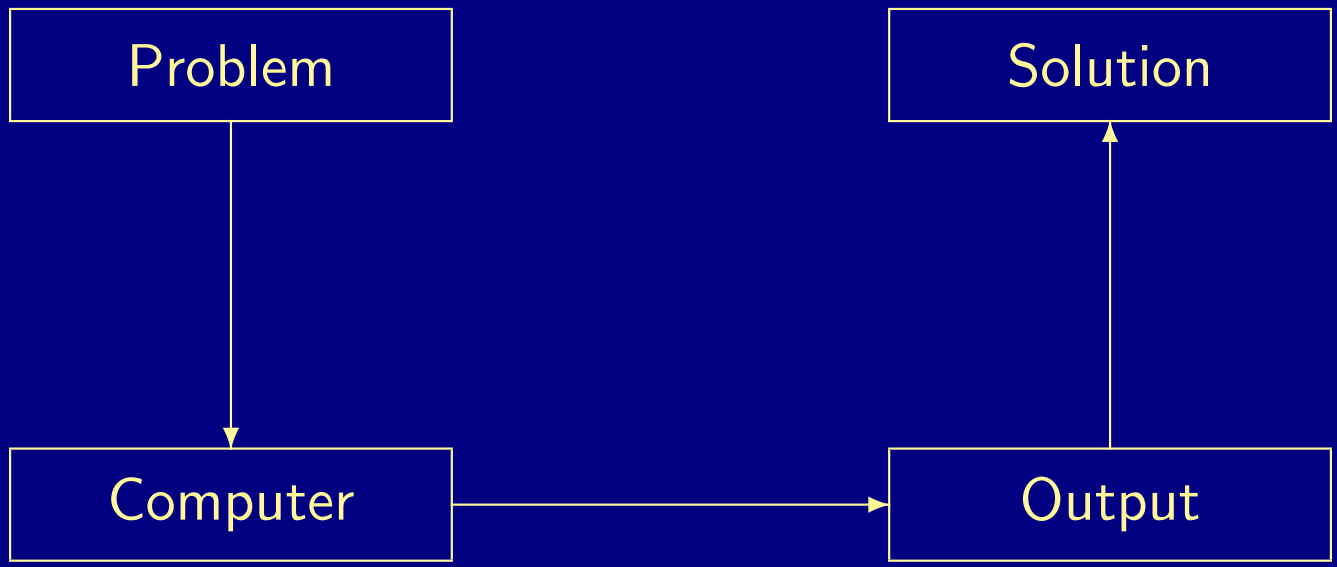
"What is the problem?" versus *"How to solve the problem?"*



Potassco

Informatics

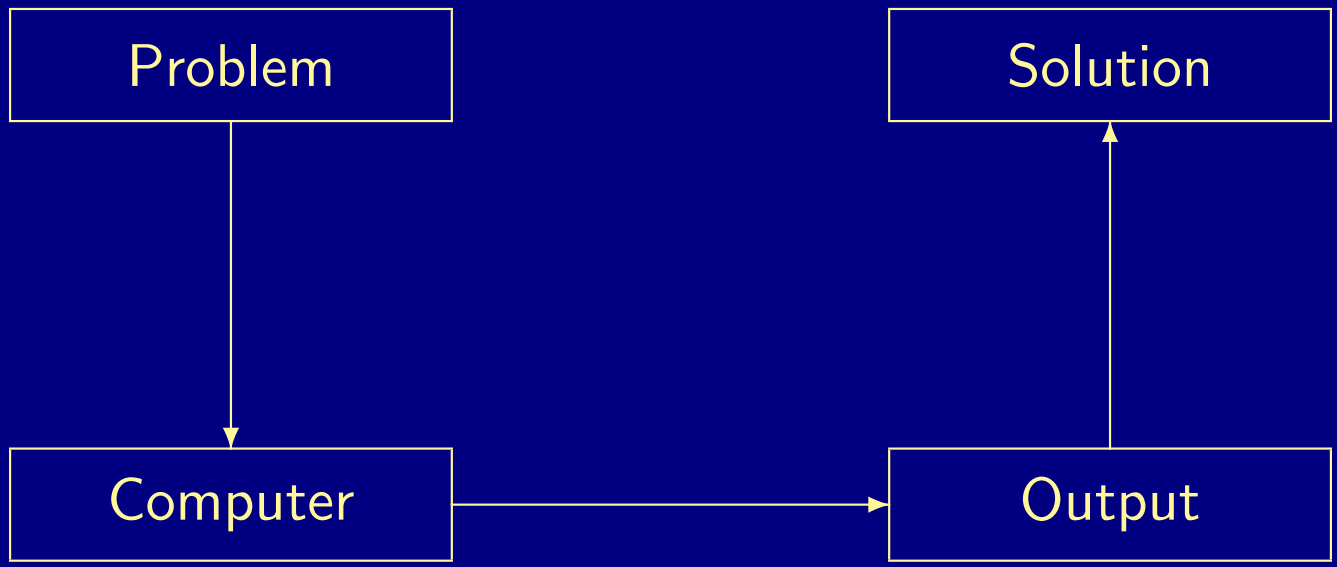
"What is the problem?" versus *"How to solve the problem?"*



Potassco

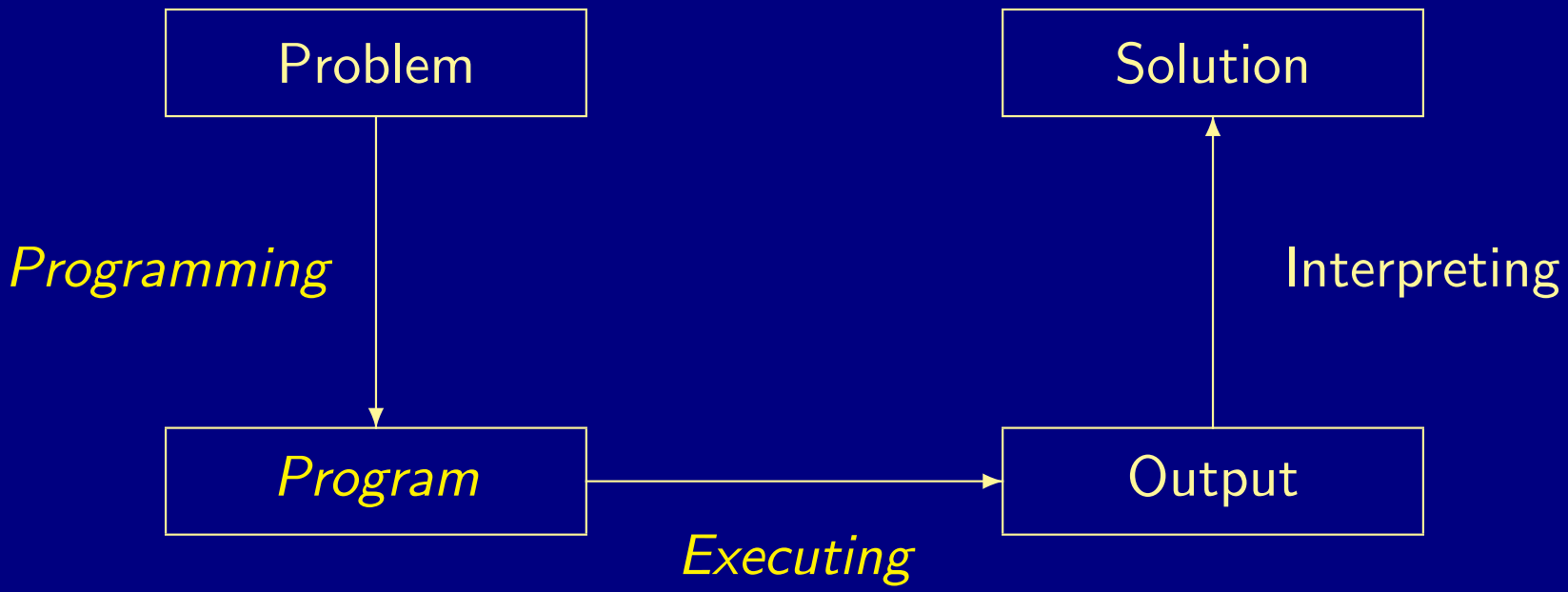
Traditional programming

"What is the problem?" versus *"How to solve the problem?"*



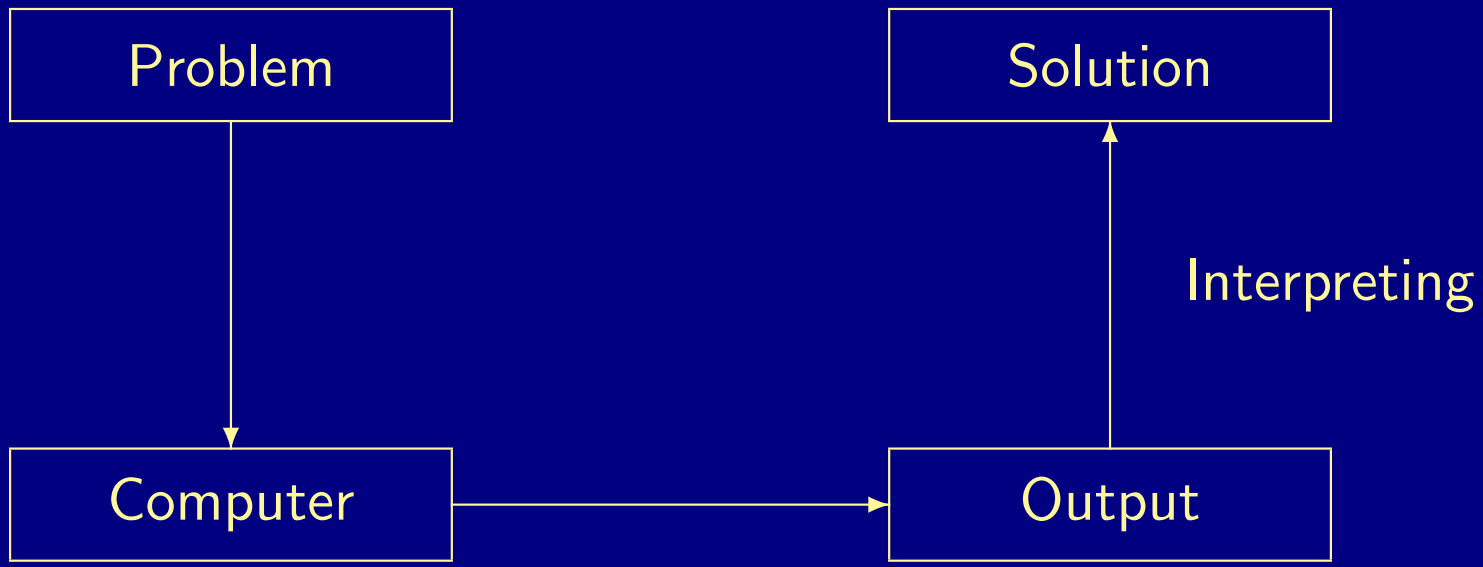
Traditional programming

"What is the problem?" versus *"How to solve the problem?"*



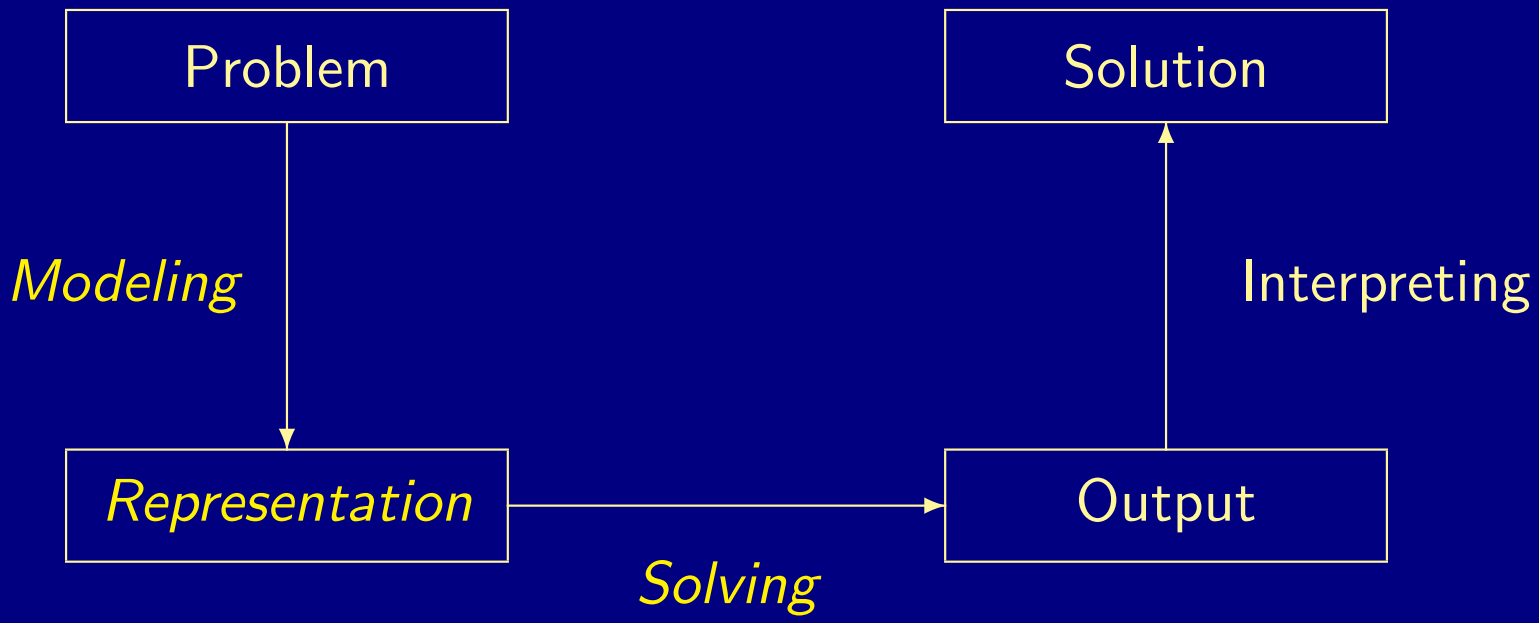
Declarative problem solving

"What is the problem?" versus *"How to solve the problem?"*



Declarative problem solving

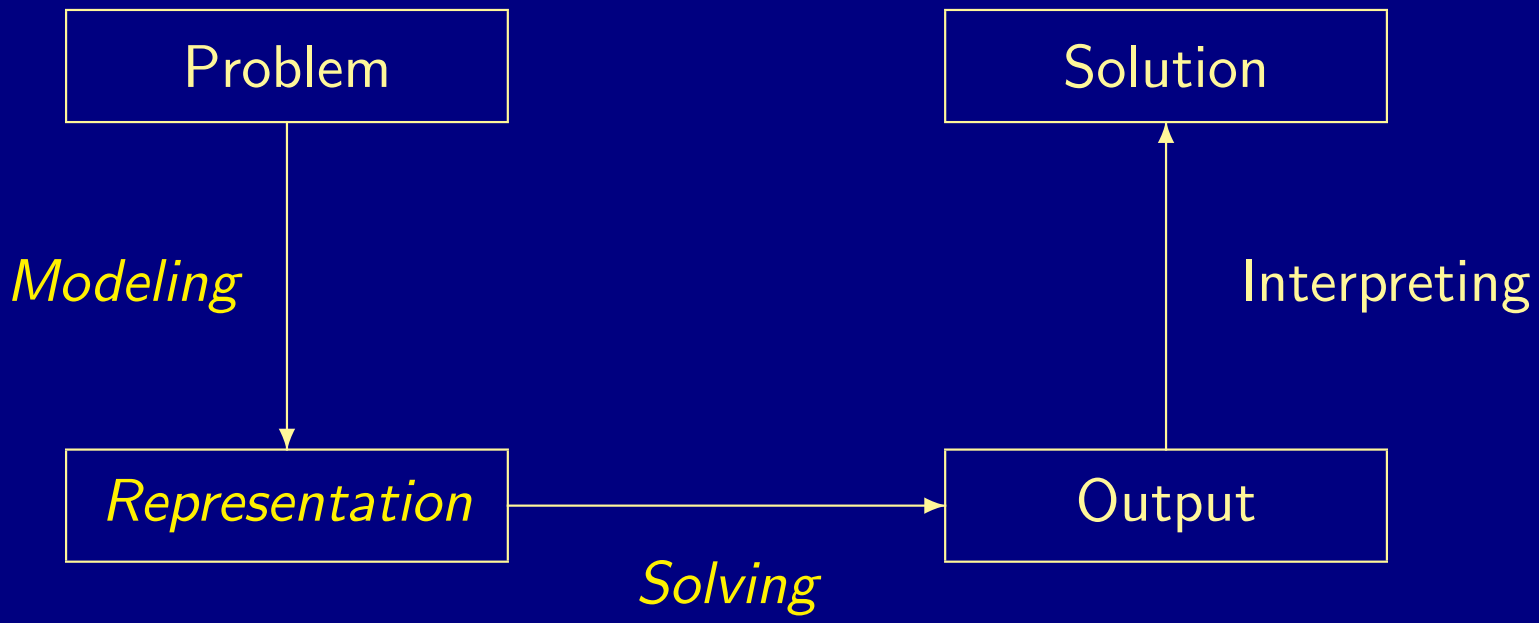
"What is the problem?" versus *"How to solve the problem?"*



Potassco

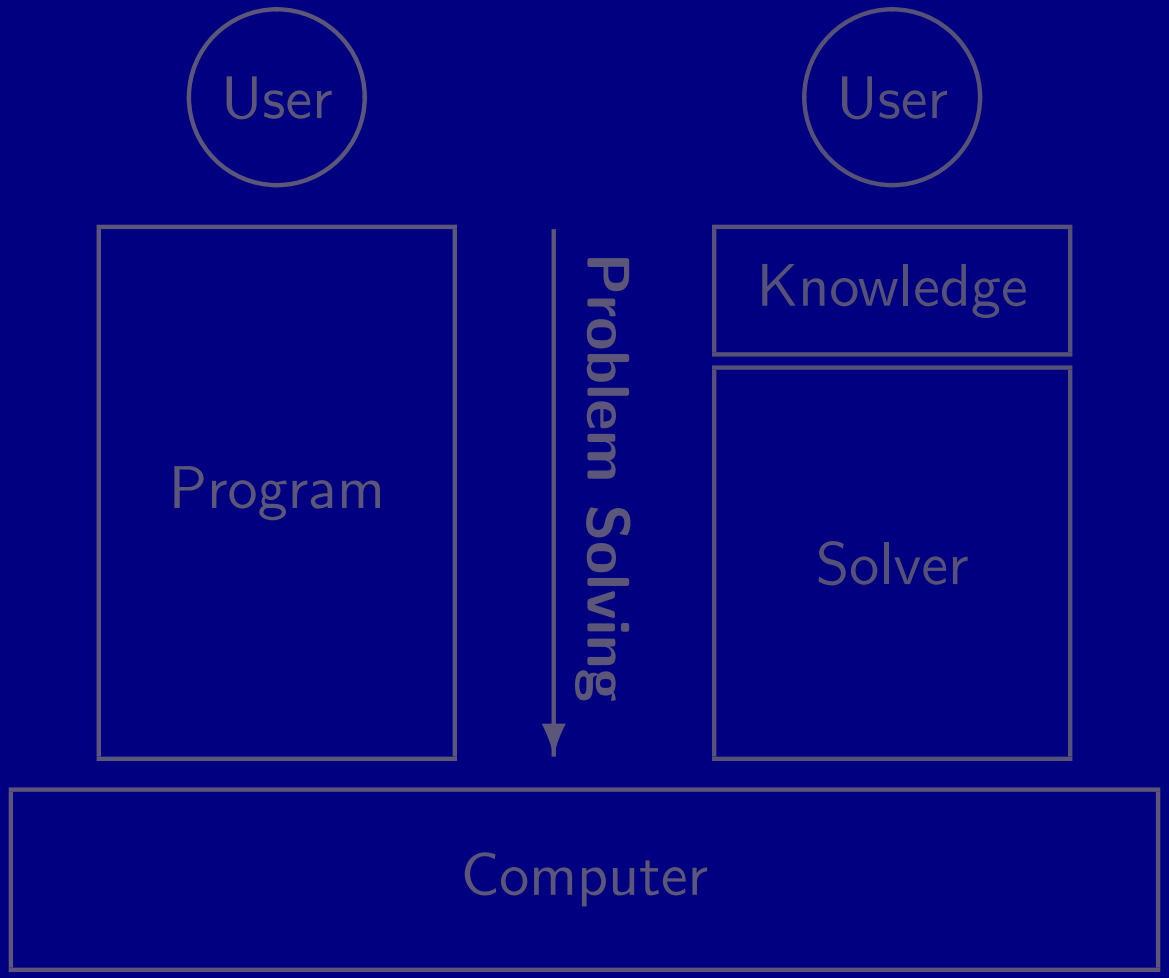
Declarative problem solving

"What is the problem?" versus *"How to solve the problem?"*



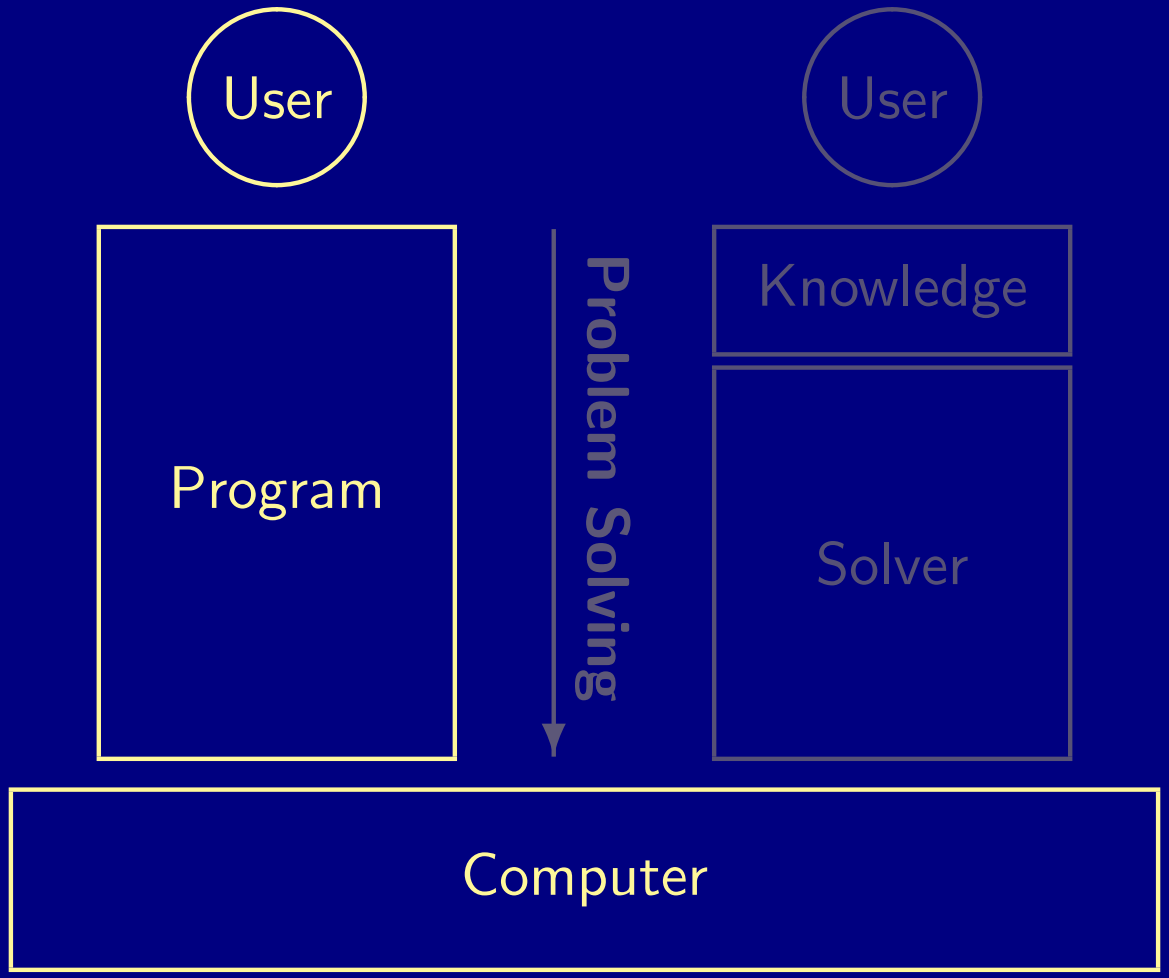
Potassco

Traditional Software

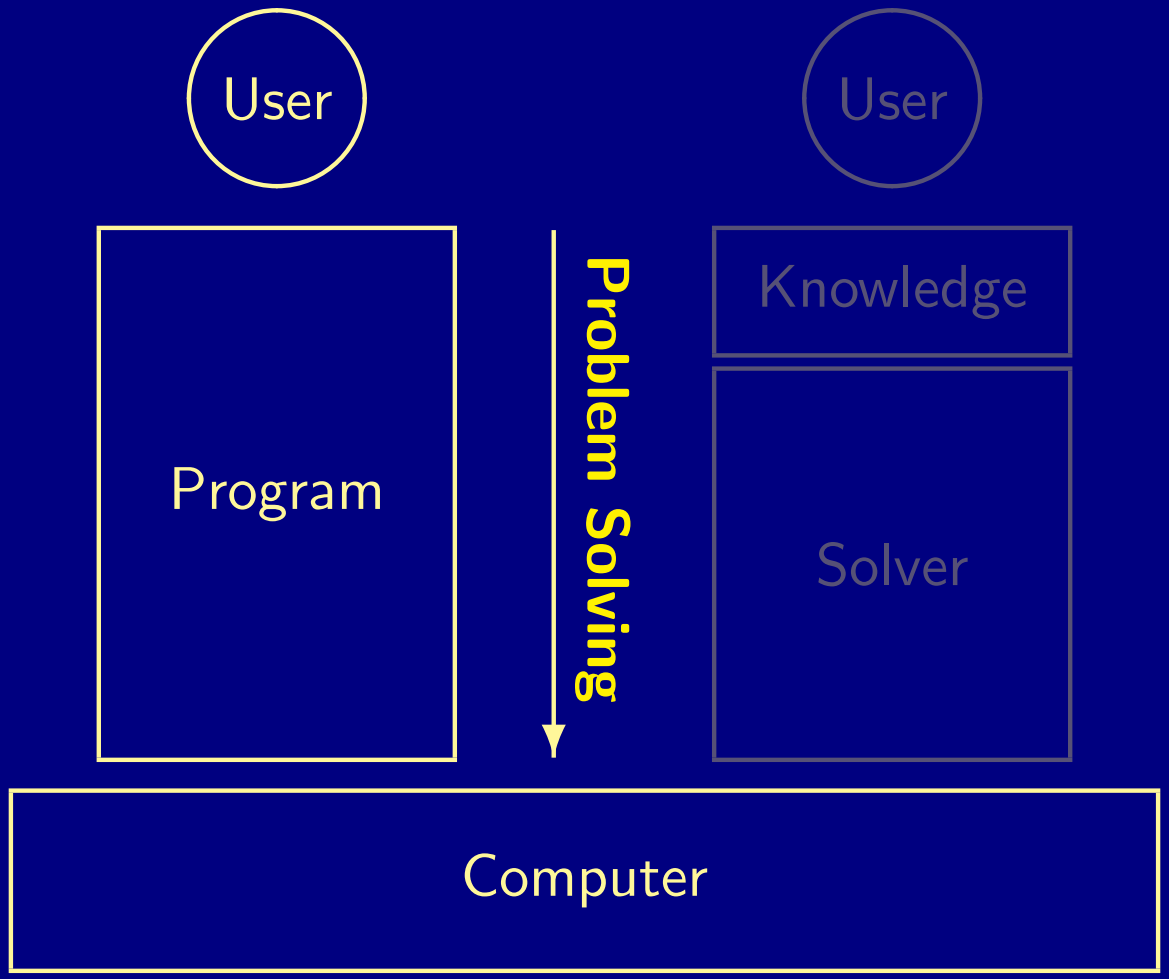


Potassco

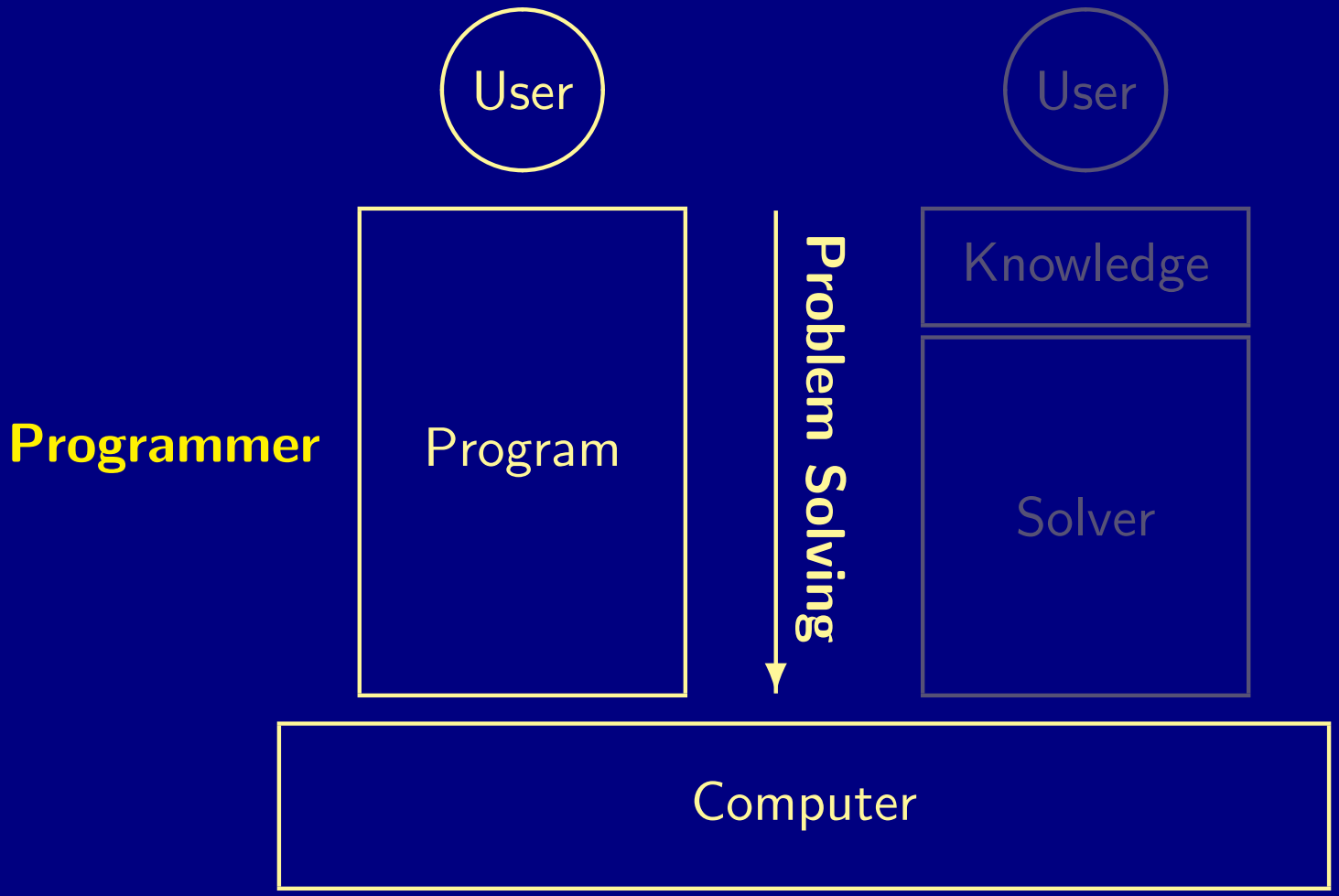
Traditional Software



Traditional Software

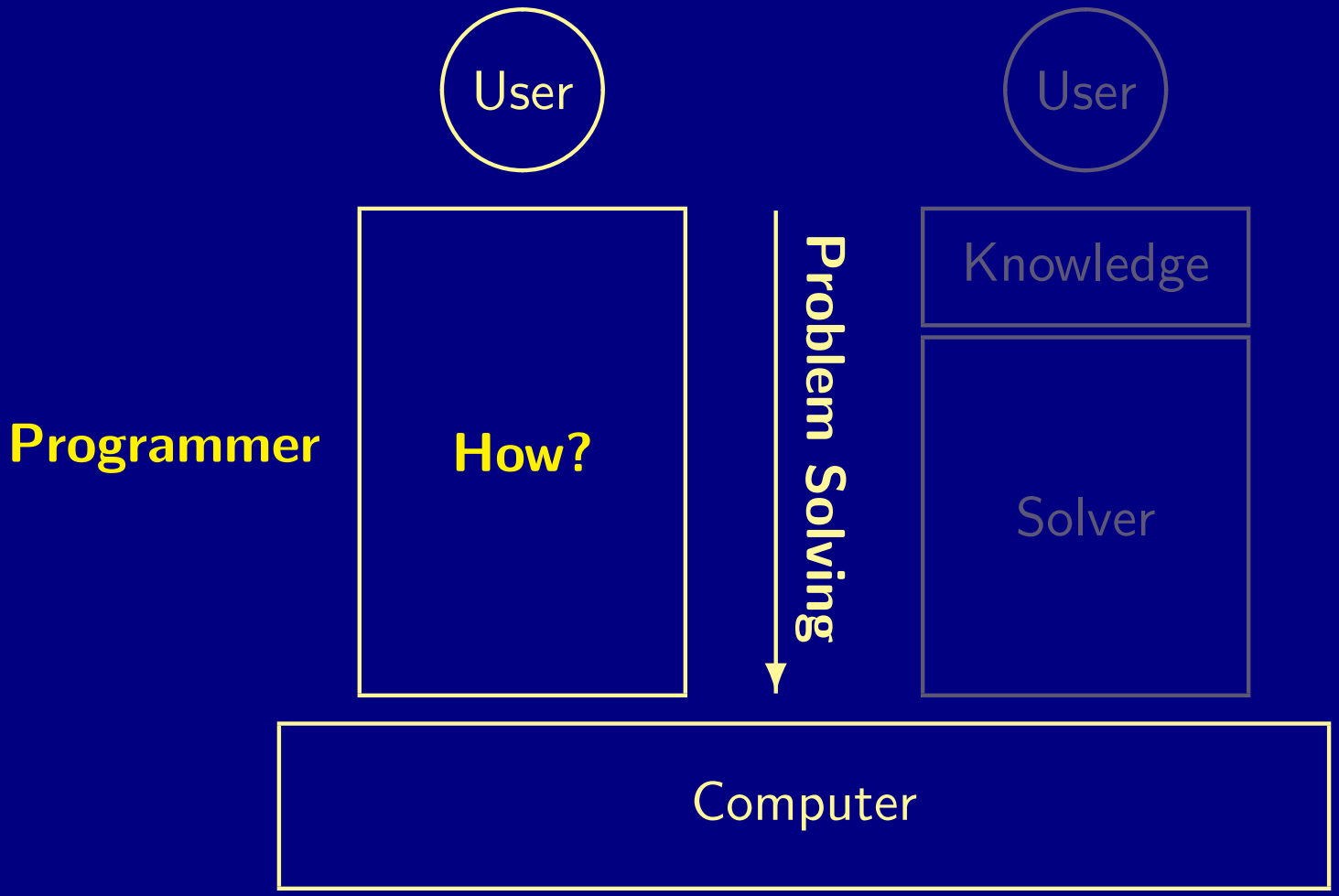


Traditional Software



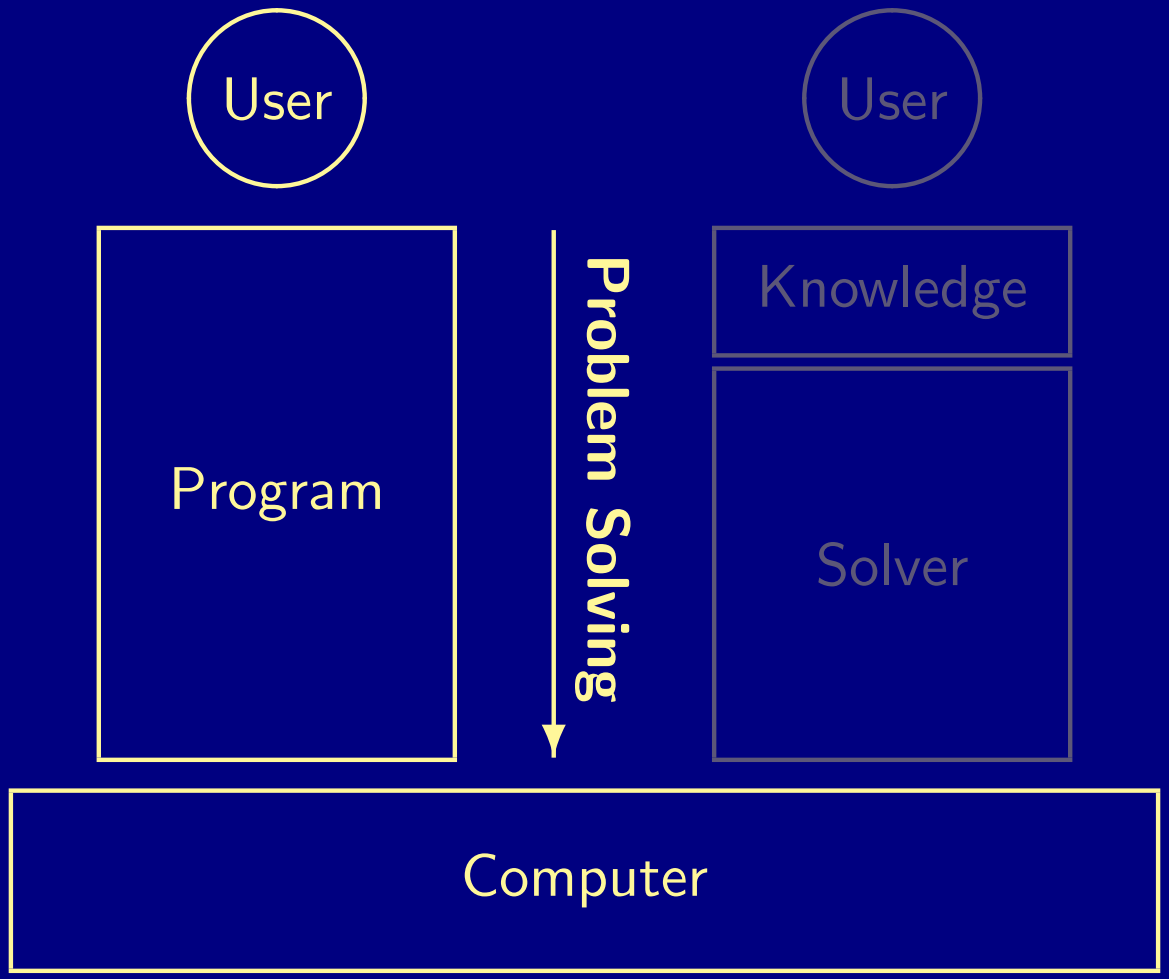
Potassco

Traditional Software



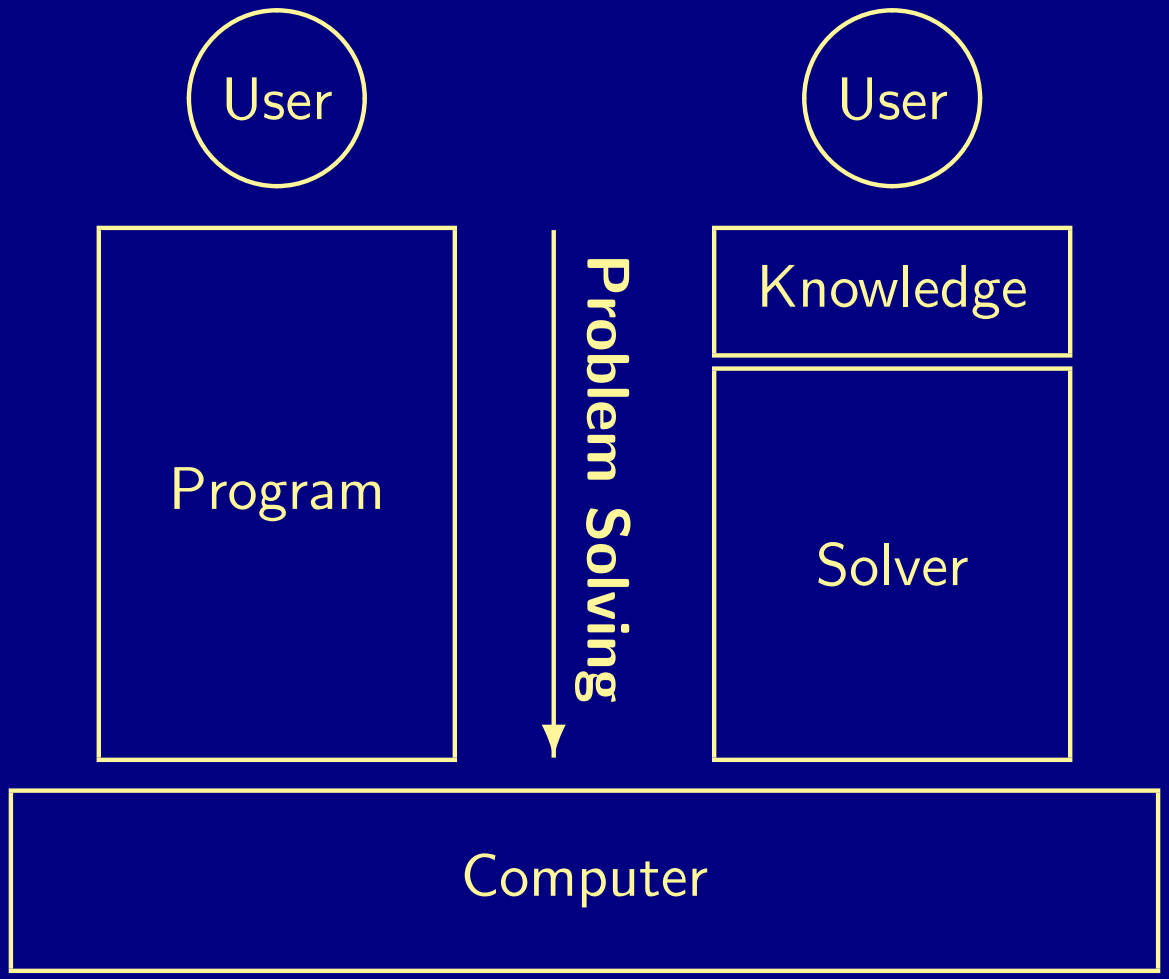
Potassco

Knowledge-driven Software



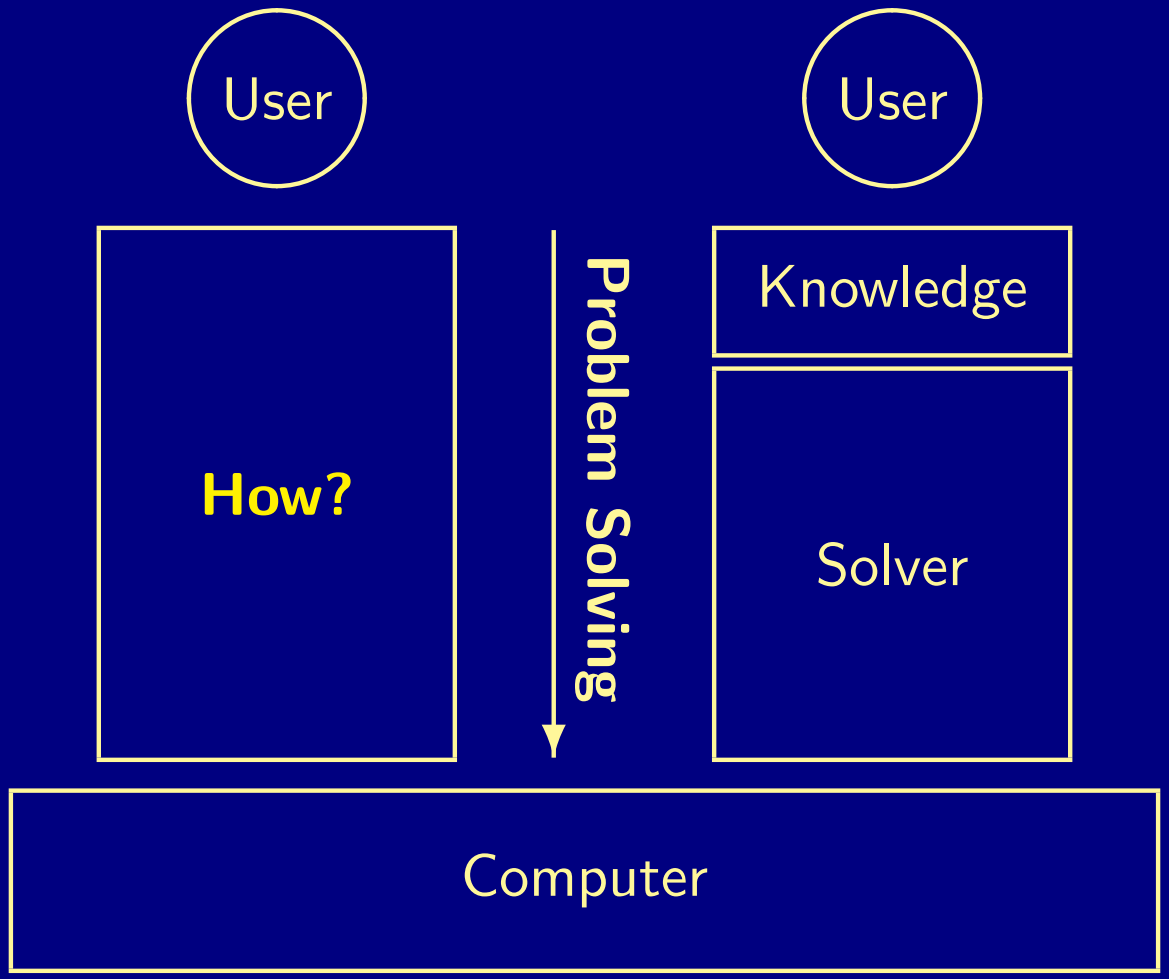
Potassco

Knowledge-driven Software



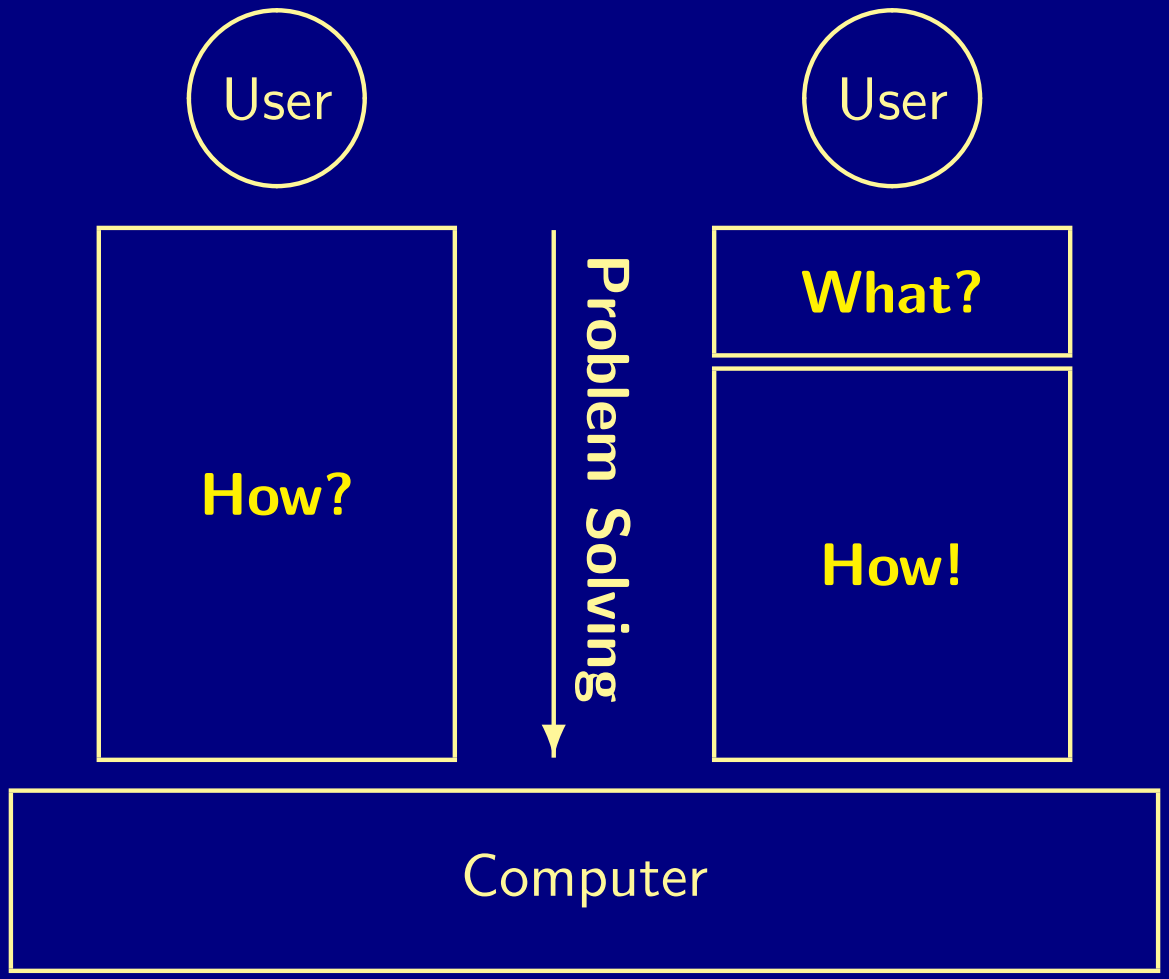
Potassco

Knowledge-driven Software

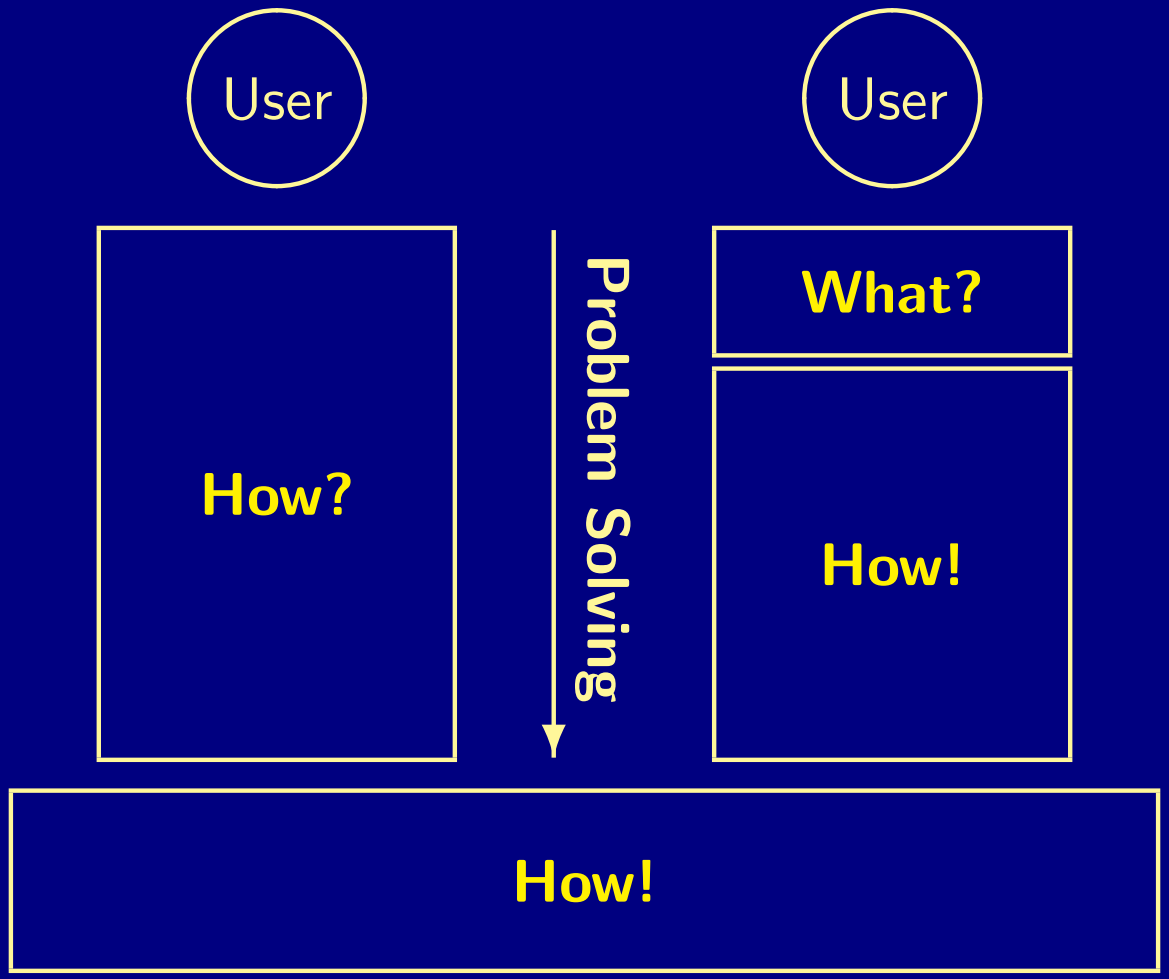


Potassco

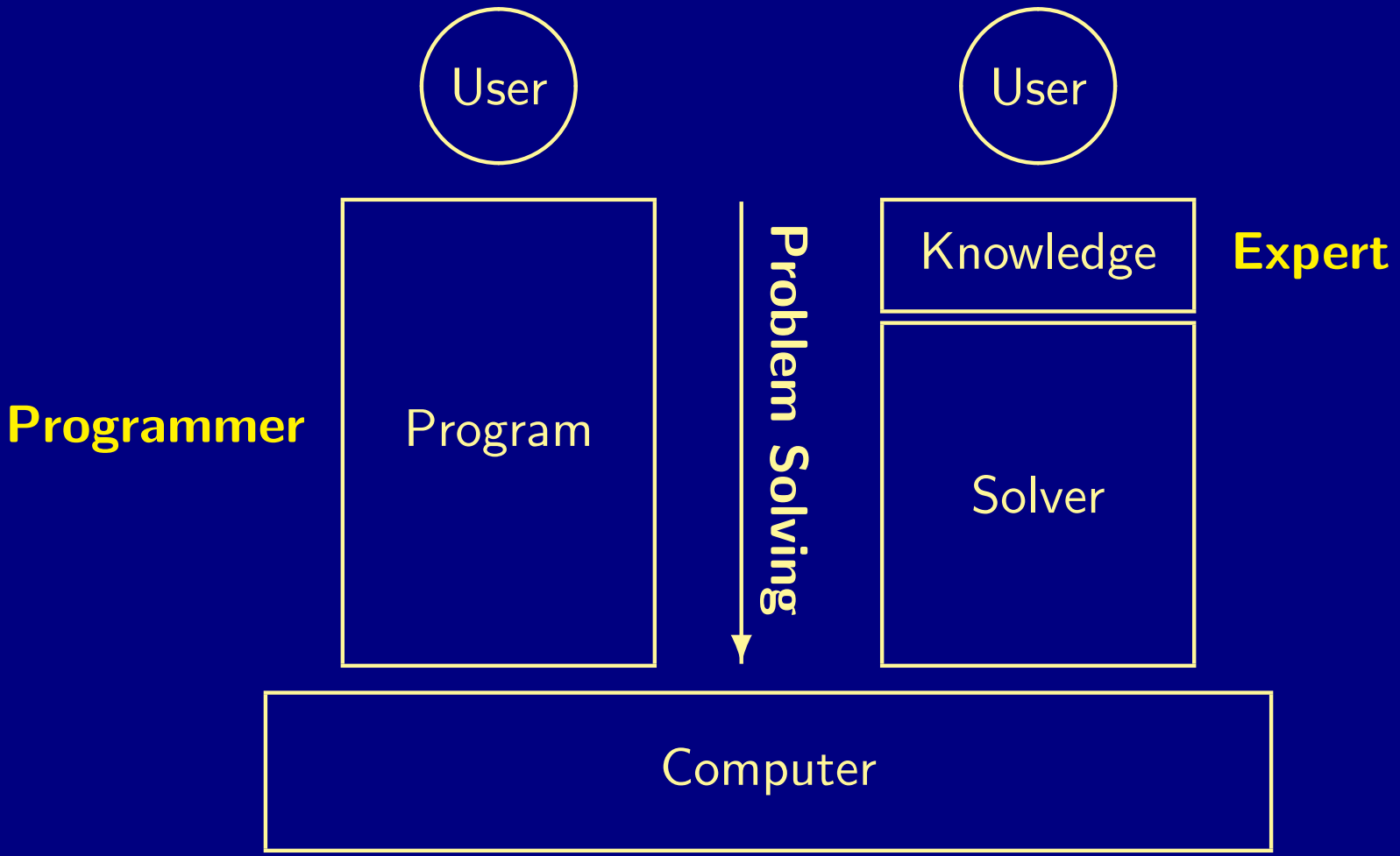
Knowledge-driven Software



Knowledge-driven Software



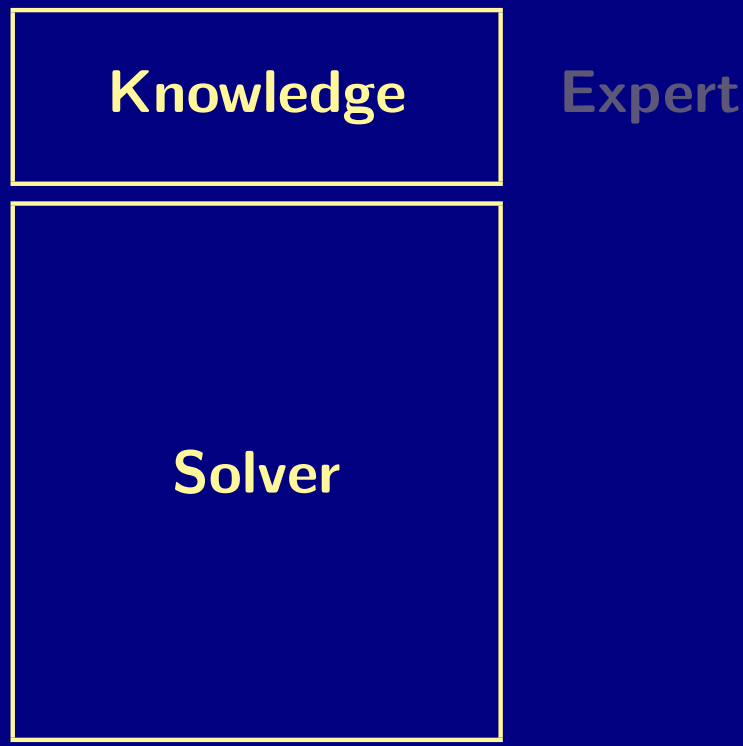
Knowledge-driven Software



What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability

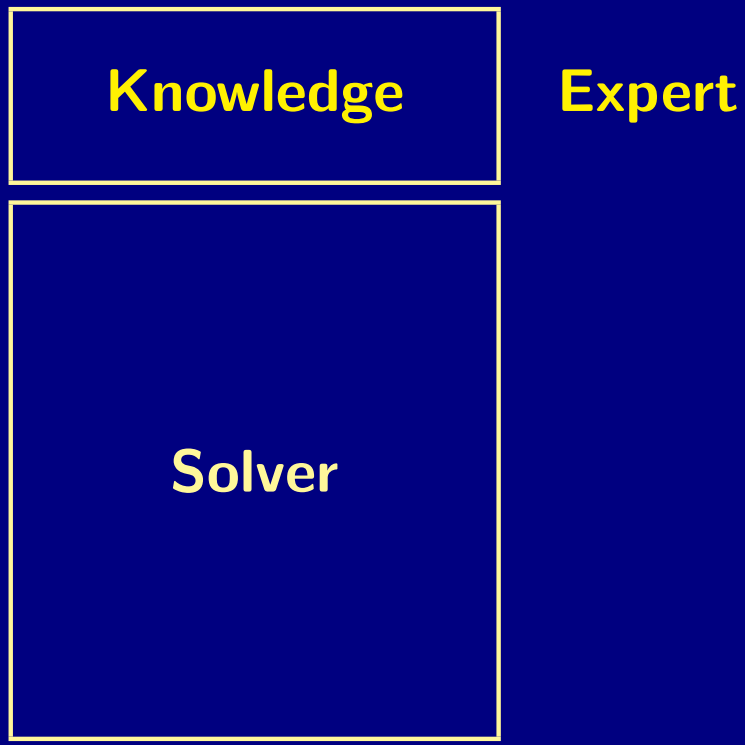


Potassco

What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability

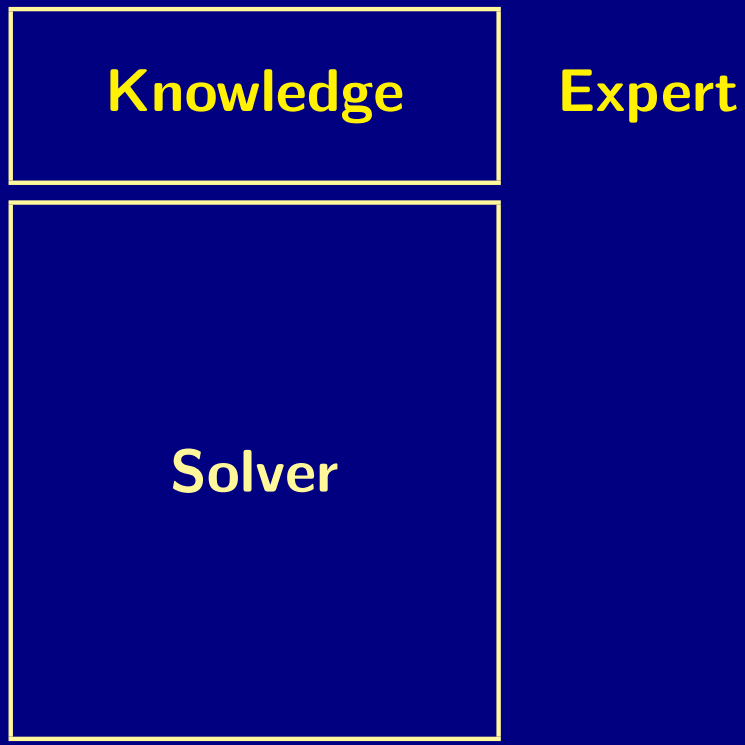


Potassco

What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability

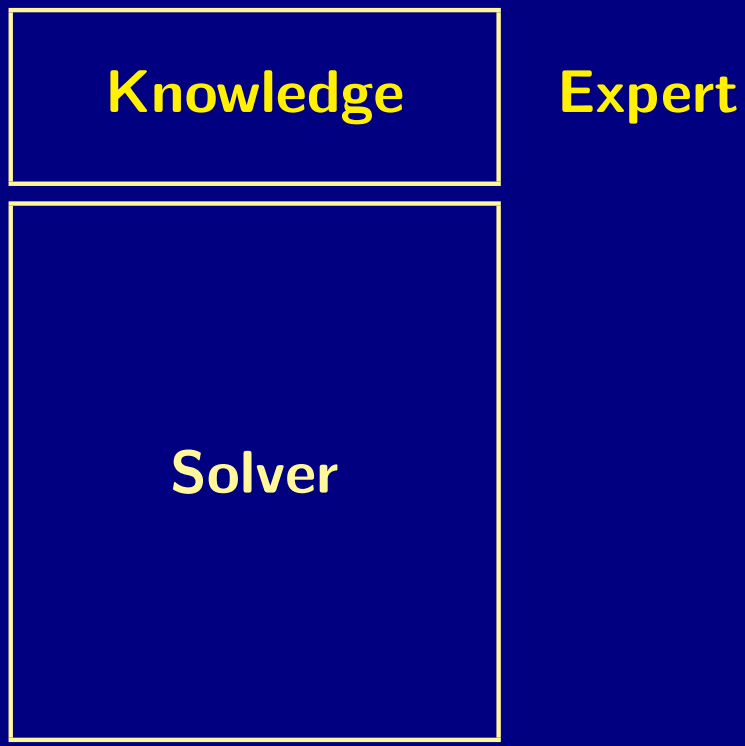


Potassco

What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability

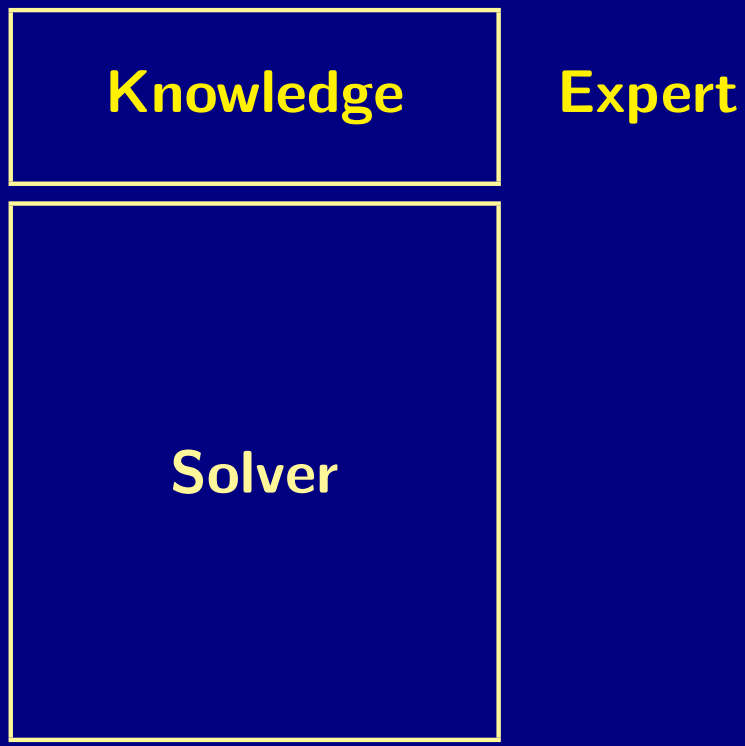


Potassco

What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability

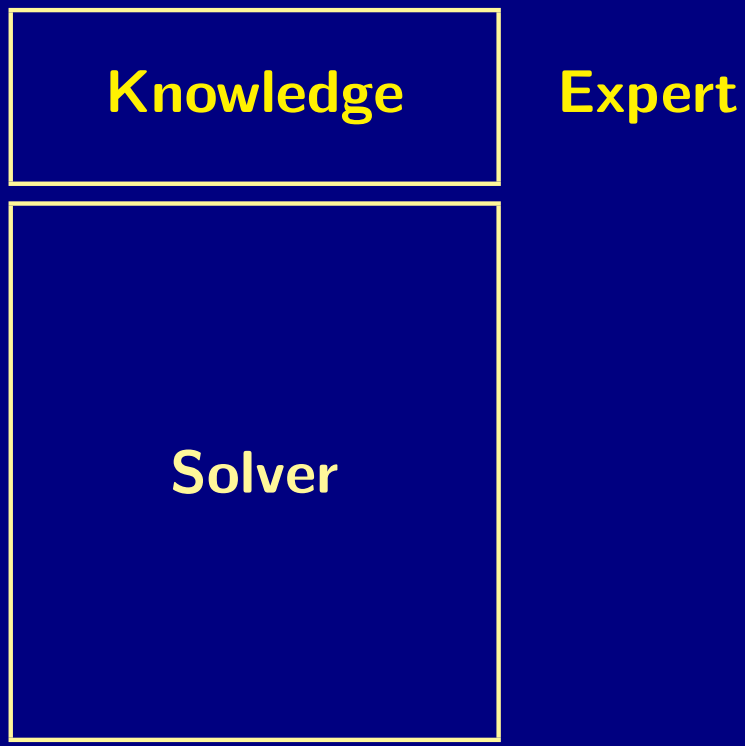


Potassco

What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability

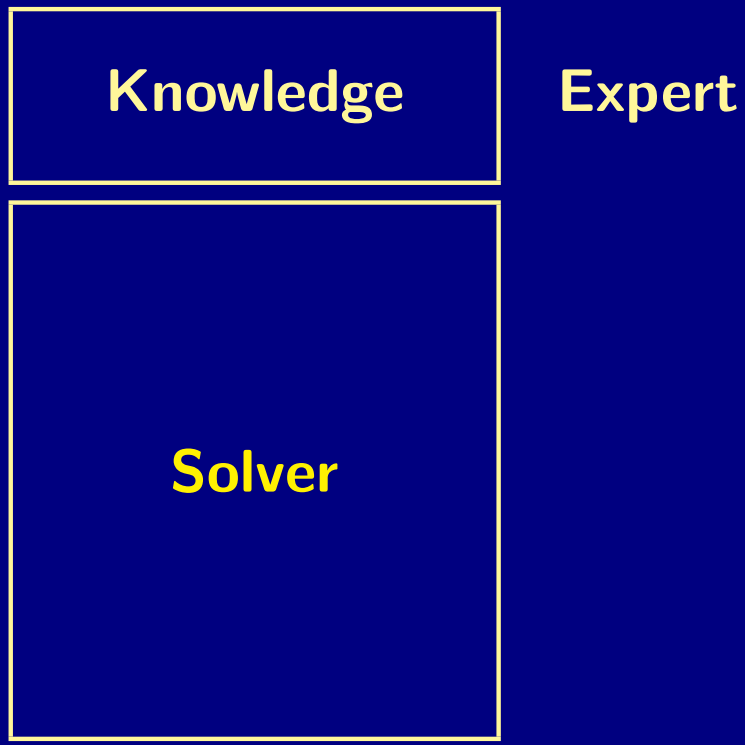


Potassco

What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability

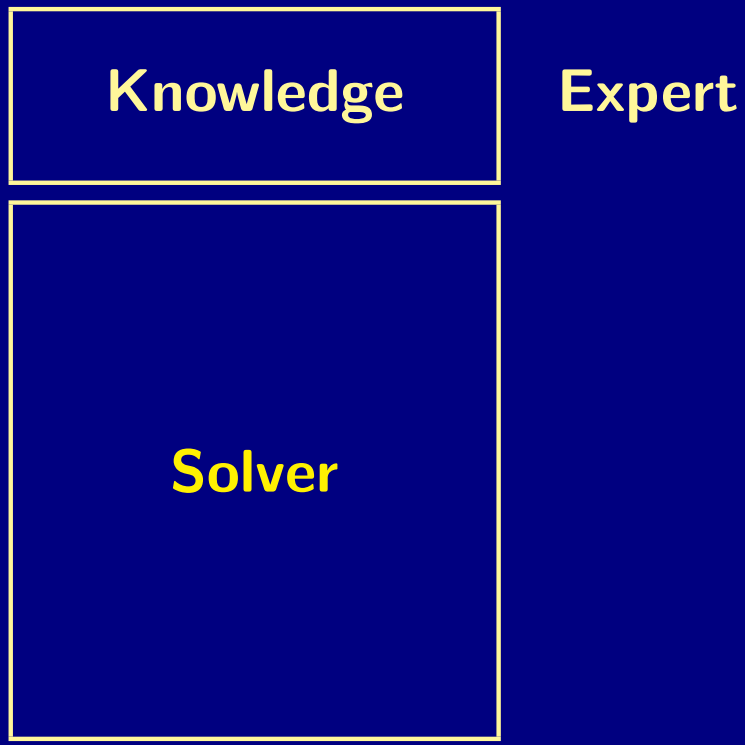


Potassco

What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability

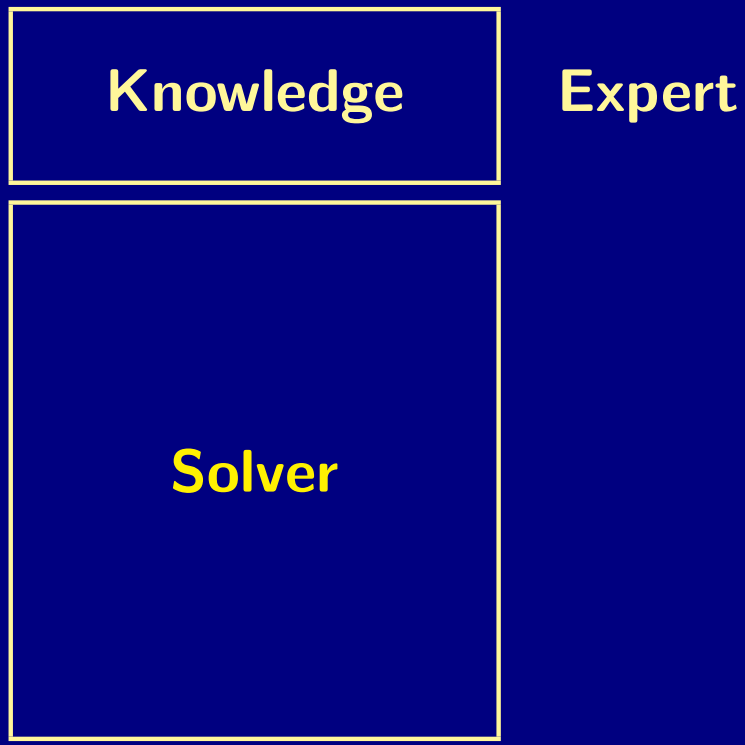


Potassco

What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability

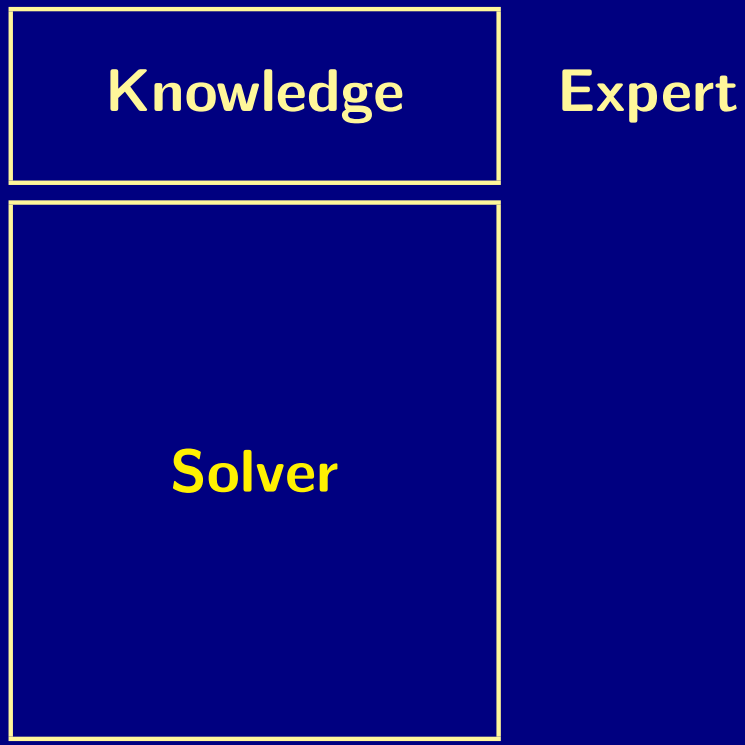


Potassco

What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability

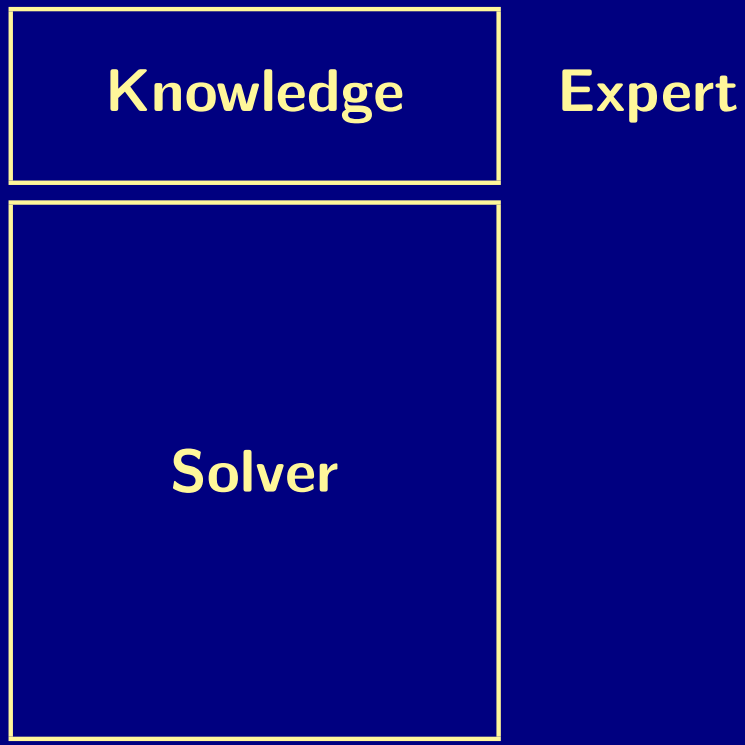


Potassco

What is the benefit?

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability



Potassco

Outline

Potassco

Answer Set Programming (ASP)

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

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

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
  - 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
  - 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 combinato
- What problems are this? — **And ind**
  - Debian, Ubuntu: Linux package cor
  - Exeura: Call routing
  - **Fcc: Radio frequency auction**
  - Gioia Tauro: Workforce management
  - Nasa: Decision support for Space Shuttle
  - 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

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
- Any industrial impact?
  - ASP Tech companies: dlv systems and potassco solutions

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
- Any industrial impact?
  - ASP Tech companies: dlv systems and potassco solutions
- Anything not so good for ASP?
  - Number crunching

Potassco

Outline

Potassco

Some biased moments in time

- '70/'80 Capturing incomplete information

Some biased moments in time

- '70/'80 Capturing incomplete information
  - Databases Closed world assumption
  - Logic programming Negation as failure
  - Non-monotonic reasoning
    Auto-epistemic and Default logics, Circumscription

Some biased moments in time

- '70/'80 Capturing incomplete information
  - Databases Closed world assumption
    - Axiomatic characterization
  - Logic programming Negation as failure
  - Non-monotonic reasoning
    Auto-epistemic and Default logics, Circumscription

Some biased moments in time

- '70/'80 Capturing incomplete information
  - Databases Closed world assumption
    - Axiomatic characterization
  - Logic programming Negation as failure
    - Herbrand interpretations
    - Fix-point characterizations
  - Non-monotonic reasoning
    Auto-epistemic and Default logics, Circumscription

Potassco

Some biased moments in time

- '70/'80 Capturing incomplete information
  - Databases Closed world assumption
    - Axiomatic characterization
  - Logic programming Negation as failure
    - Herbrand interpretations
    - Fix-point characterizations
  - Non-monotonic reasoning
    Auto-epistemic and Default logics, Circumscription
    - Extensions of first-order logic
    - Modalities, fix-points, second-order logic

Potassco

Some biased moments in time

- '70/'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

Potassco

Some biased moments in time

- '70/'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
  - Logic programming semantics
    Well-founded and stable models semantics
  - ASP solving
    "Stable models = Well-founded semantics + Branch"

Potassco

Some biased moments in time

- '70/'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
  - Logic programming semantics
    Well-founded and stable models semantics
    - Stable models semantics derived from non-monotonic logics
    - Alternating fix-point theory
  - ASP solving
    "Stable models = Well-founded semantics + Branch"

Potassco

Some biased moments in time

- '70/'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
  - Logic programming semantics
    Well-founded and stable models semantics
    - Stable models semantics derived from non-monotonic logics
    - Alternating fix-point theory
  - ASP solving
    "Stable models = Well-founded semantics + Branch"
    - Modeling — Grounding — Solving
    - Icebreakers: `lparse` and `smodels`

Potassco

Some biased moments in time

- '70/'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
  - Logic programming semantics
    Well-founded and stable models semantics
  - ASP solving
    "Stable models = Well-founded semantics + Branch"
- '00 Applications and semantic rediscoveries

Potassco

Some biased moments in time

- '70/'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
  - Logic programming semantics
    Well-founded and stable models semantics
  - ASP solving
    "Stable models = Well-founded semantics + Branch"
- '00 Applications and semantic rediscoveries
  - Growing dissemination Decision Support for Space Shuttle
  - Constructive logics Equilibrium Logic

Potassco

Some biased moments in time

- '70/'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
  - Logic programming semantics
    Well-founded and stable models semantics
  - ASP solving
    "Stable models = Well-founded semantics + Branch"
- '00 Applications and semantic rediscoveries
  - Growing dissemination Decision Support for Space Shuttle
    - Bio-informatics, Linux Package Configuration, Music composition, Robotics, System Design, etc
  - Constructive logics Equilibrium Logic

Potassco

Some biased moments in time

- '70/'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
  - Logic programming semantics
    Well-founded and stable models semantics
  - ASP solving
    "Stable models = Well-founded semantics + Branch"
- '00 Applications and semantic rediscoveries
  - Growing dissemination Decision Support for Space Shuttle
  - Constructive logics Equilibrium Logic
    - Roots: Logic of Here-and-There , G3

Potassco

Some biased moments in time

- '70/'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
  - Logic programming semantics
    Well-founded and stable models semantics
  - ASP solving
    "Stable models = Well-founded semantics + Branch"
- '00 Applications and semantic rediscoveries
  - Growing dissemination Decision Support for Space Shuttle
  - Constructive logics Equilibrium Logic
- '10 Integration

Potassco

Some biased moments in time

- '70/'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
  - Logic programming semantics
    Well-founded and stable models semantics
  - ASP solving
    "Stable models = Well-founded semantics + Branch"
- '00 Applications and semantic rediscoveries
  - Growing dissemination Decision Support for Space Shuttle
  - Constructive logics Equilibrium Logic
- '10 Integration — let's see . . .

Potassco

Some biased moments in time

- '70/'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
  - Logic programming semantics
    Well-founded and stable models semantics
  - ASP solving
    "Stable models = Well-founded semantics + Branch"
- '00 Applications and semantic rediscoveries
  - Growing dissemination Decision Support for Space Shuttle
  - Constructive logics Equilibrium Logic
- '10 Integration

Potassco

Paradigm shift

Theorem Proving based approach (eg. Prolog)

1. Provide a representation of the problem
2. A solution is given by a derivation of a query

Model Generation based approach (eg. SATisfiability testing)

1. Provide a representation of the problem
2. A solution is given by a model of the representation

Automated planning, Kautz and Selman (ECAI'92)

Represent planning problems as propositional theories so that models not proofs describe solutions

Potassco

Paradigm shift

Theorem Proving based approach (eg. Prolog)

1. Provide a representation of the problem
2. A solution is given by a derivation of a query

Model Generation based approach (eg. SATisfiability testing)

1. Provide a representation of the problem
2. A solution is given by a model of the representation

Automated planning, Kautz and Selman (ECAI'92)

Represent planning problems as propositional theories so that models not proofs describe solutions

Potassco

Paradigm shift

Theorem Proving based approach (eg. Prolog)

1. Provide a representation of the problem
2. A solution is given by a derivation of a query

Model Generation based approach (eg. SATisfiability testing)

1. Provide a representation of the problem
2. A solution is given by a model of the representation

Automated planning, Kautz and Selman (ECAI'92)

Represent planning problems as propositional theories so that models not proofs describe solutions

Potassco

Paradigm shift

Theorem Proving based approach (eg. Prolog)

1. Provide a representation of the problem
2. A solution is given by a derivation of a query

Model Generation based approach (eg. SATisfiability testing)

1. Provide a representation of the problem
2. A solution is given by a model of the representation

Automated planning, Kautz and Selman (ECAI'92)

Represent planning problems as propositional theories so that models not proofs describe solutions

Potassco

Model Generation based Problem Solving

Representation	Solution
constraint satisfaction problem	assignment
propositional horn theories	smallest model
propositional theories	models
propositional theories	minimal models
propositional theories	stable models
propositional programs	minimal models
propositional programs	supported models
propositional programs	stable models
first-order theories	models
first-order theories	minimal models
first-order theories	stable models
first-order theories	Herbrand models
auto-epistemic theories	expansions
default theories	extensions
⋮	⋮

Potassco

Model Generation based Problem Solving

Representation	Solution
constraint satisfaction problem	assignment
propositional horn theories	smallest model
propositional theories	models
propositional theories	minimal models
propositional theories	stable models
propositional programs	minimal models
propositional programs	supported models
propositional programs	stable models
first-order theories	models
first-order theories	minimal models
first-order theories	stable models
first-order theories	Herbrand models
auto-epistemic theories	expansions
default theories	extensions
⋮	⋮

Potassco

Model Generation based Problem Solving

Representation	Solution	
constraint satisfaction problem	assignment	
propositional horn theories	smallest model	
propositional theories	models	**SAT**
propositional theories	minimal models	
propositional theories	stable models	
propositional programs	minimal models	
propositional programs	supported models	
propositional programs	stable models	
first-order theories	models	
first-order theories	minimal models	
first-order theories	stable models	
first-order theories	Herbrand models	
auto-epistemic theories	expansions	
default theories	extensions	
⋮	⋮	

Potassco

Paradigm shift

Theorem Proving based approach (eg. Prolog)

1. Provide a representation of the problem
2. A solution is given by a derivation of a query

Model Generation based approach (eg. SATisfiability testing)

1. Provide a representation of the problem
2. A solution is given by a model of the representation

Potassco

Paradigm shift

Theorem Proving based approach (eg. Prolog)

1. Provide a representation of the problem
2. A solution is given by a derivation of a query

Model Generation based approach (eg. SATisfiability testing)

1. Provide a representation of the problem
2. A solution is given by a model of the representation

LP-style playing with blocks

Prolog program

```
on(a,b).
on(b,c).

above(X,Y) :- on(X,Y).
above(X,Y) :- on(X,Z), above(Z,Y).
```

Prolog queries

```
?- above(a,c).
true.

?- above(c,a).
no.
```

Potassco

LP-style playing with blocks

Prolog program

```
on(a,b).
on(b,c).

above(X,Y) :- on(X,Y).
above(X,Y) :- on(X,Z), above(Z,Y).
```

Prolog queries

```
?- above(a,c).
true.

?- above(c,a).
no.
```

Potassco

LP-style playing with blocks

Prolog program

```
on(a,b).
on(b,c).

above(X,Y) :- on(X,Y).
above(X,Y) :- on(X,Z), above(Z,Y).
```

Prolog queries

```
?- above(a,c).
true.

?- above(c,a).
no.
```

Potassco

LP-style playing with blocks

Prolog program

```
on(a,b).
on(b,c).

above(X,Y) :- on(X,Y).
above(X,Y) :- on(X,Z), above(Z,Y).
```

Prolog queries (testing entailment)

```
?- above(a,c).
true.

?- above(c,a).
no.
```

Potassco

LP-style playing with blocks

Shuffled Prolog program

```
on(a,b).
on(b,c).

above(X,Y) :- above(X,Z), on(Z,Y).
above(X,Y) :- on(X,Y).
```

Prolog queries

```
?- above(a,c).

Fatal Error: local stack overflow.
```

Potassco

LP-style playing with blocks

Shuffled Prolog program

```
on(a,b).
on(b,c).

above(X,Y) :- above(X,Z), on(Z,Y).
above(X,Y) :- on(X,Y).
```

Prolog queries

```
?- above(a,c).

Fatal Error: local stack overflow.
```

Potassco

LP-style playing with blocks

Shuffled Prolog program

```
on(a,b).
on(b,c).

above(X,Y) :- above(X,Z), on(Z,Y).
above(X,Y) :- on(X,Y).
```

Prolog queries (answered via fixed execution)

```
?- above(a,c).

Fatal Error: local stack overflow.
```

Potassco

Paradigm shift

Theorem Proving based approach (eg. Prolog)

1. Provide a representation of the problem
2. A solution is given by a derivation of a query

Model Generation based approach (eg. SATisfiability testing)

1. Provide a representation of the problem
2. A solution is given by a model of the representation

Potassco

Paradigm shift

Theorem Proving based approach (eg. Prolog)

1. Provide a representation of the problem
2. A solution is given by a derivation of a query

Model Generation based approach (eg. SATisfiability testing)

1. Provide a representation of the problem
2. A solution is given by a model of the representation

SAT-style playing with blocks

Formula

$$
\begin{array}{ll}
 & on(a,b) \\
\wedge & on(b,c) \\
\wedge & (on(X,Y) \rightarrow above(X,Y)) \\
\wedge & (on(X,Z) \wedge above(Z,Y) \rightarrow above(X,Y))
\end{array}
$$

Herbrand model

$$
\left\{
\begin{array}{ccccccc}
on(a,b), & & on(b,c), & & on(a,c), & on(b,b), & \\
above(a,b), & above(b,c), & & above(a,c), & above(b,b), & above(c,b)
\end{array}
\right\}
$$

SAT-style playing with blocks

Formula

$$
\begin{array}{ll}
 & on(a,b) \\
\wedge & on(b,c) \\
\wedge & (on(X,Y) \rightarrow above(X,Y)) \\
\wedge & (on(X,Z) \wedge above(Z,Y) \rightarrow above(X,Y))
\end{array}
$$

Herbrand model

$$
\left\{
\begin{array}{ccccccccc}
on(a,b), & & on(b,c), & & on(a,c), & & on(b,b), & & \\
above(a,b), & & above(b,c), & & above(a,c), & & above(b,b), & & above(c,b)
\end{array}
\right\}
$$

SAT-style playing with blocks

Formula

$$
\begin{array}{ll}
 & on(a,b) \\
\wedge & on(b,c) \\
\wedge & (on(X,Y) \rightarrow above(X,Y)) \\
\wedge & (on(X,Z) \wedge above(Z,Y) \rightarrow above(X,Y))
\end{array}
$$

Herbrand model

$$
\left\{
\begin{array}{ccccc}
on(a,b), & on(b,c), & on(a,c), & on(b,b), & \\
above(a,b), & above(b,c), & above(a,c), & above(b,b), & above(c,b)
\end{array}
\right\}
$$

SAT-style playing with blocks

Formula

$$
\begin{array}{ll}
 & on(a,b) \\
\wedge & on(b,c) \\
\wedge & (on(X,Y) \rightarrow above(X,Y)) \\
\wedge & (on(X,Z) \wedge above(Z,Y) \rightarrow above(X,Y))
\end{array}
$$

Herbrand model

$$
\left\{
\begin{array}{ccccccc}
on(a,b), & on(b,c), & on(a,c), & on(b,b), & \\
above(a,b), & above(b,c), & above(a,c), & above(b,b), & above(c,b)
\end{array}
\right\}
$$

SAT-style playing with blocks

Formula

$$
\begin{array}{ll}
 & on(a,b) \\
\wedge & on(b,c) \\
\wedge & (on(X,Y) \rightarrow above(X,Y)) \\
\wedge & (on(X,Z) \wedge above(Z,Y) \rightarrow above(X,Y))
\end{array}
$$

Herbrand model (among 426!)

$$
\left\{
\begin{array}{ccccccccc}
on(a,b), & & on(b,c), & & on(a,c), & & on(b,b), & & \\
above(a,b), & above(b,c), & & above(a,c), & & above(b,b), & & above(c,b) &
\end{array}
\right\}
$$

Paradigm shift

Theorem Proving based approach (eg. Prolog)

1. Provide a representation of the problem
2. A solution is given by a derivation of a query

Model Generation based approach (eg. SATisfiability testing)

1. Provide a representation of the problem
2. A solution is given by a model of the representation

Potassco

Paradigm shift

Theorem Proving based approach (eg. Prolog)

1. Provide a representation of the problem
2. A solution is given by a derivation of a query

Model Generation based approach (eg. SATisfiability testing)

1. Provide a representation of the problem
2. A solution is given by a model of the representation

➥ Answer Set Programming (ASP)

Potassco

Model Generation based Problem Solving

Representation	Solution
constraint satisfaction problem	assignment
propositional horn theories	smallest model
propositional theories	models
propositional theories	minimal models
propositional theories	stable models
propositional programs	minimal models
propositional programs	supported models
propositional programs	stable models
first-order theories	models
first-order theories	minimal models
first-order theories	stable models
first-order theories	Herbrand models
auto-epistemic theories	expansions
default theories	extensions
⋮	⋮

Potassco

Answer Set Programming *at large*

Representation	Solution
constraint satisfaction problem	assignment
propositional horn theories	smallest model
propositional theories	models
propositional theories	minimal models
propositional theories	stable models
propositional programs	minimal models
propositional programs	supported models
propositional programs	stable models
first-order theories	models
first-order theories	minimal models
first-order theories	stable models
first-order theories	Herbrand models
auto-epistemic theories	expansions
default theories	extensions
⋮	⋮

Potassco

Answer Set Programming *commonly*

Representation	Solution
constraint satisfaction problem	assignment
propositional horn theories	smallest model
propositional theories	models
propositional theories	minimal models
propositional theories	stable models
propositional programs	minimal models
propositional programs	supported models
propositional programs	stable models
first-order theories	models
first-order theories	minimal models
first-order theories	stable models
first-order theories	Herbrand models
auto-epistemic theories	expansions
default theories	extensions
⋮	⋮

Potassco

Answer Set Programming *in practice*

Representation	Solution
constraint satisfaction problem	assignment
propositional horn theories	smallest model
propositional theories	models
propositional theories	minimal models
propositional theories	stable models
propositional programs	minimal models
propositional programs	supported models
propositional programs	stable models
first-order theories	models
first-order theories	minimal models
first-order theories	stable models
first-order theories	Herbrand models
auto-epistemic theories	expansions
default theories	extensions
⋮	⋮

Potassco

Answer Set Programming *in practice*

Representation	Solution
constraint satisfaction problem	assignment
propositional horn theories	smallest model
propositional theories	models
propositional theories	minimal models
propositional theories	stable models
propositional programs	minimal models
propositional programs	supported models
propositional programs	stable models
first-order theories	models
first-order theories	minimal models
first-order theories	stable models
first-order theories	Herbrand models
auto-epistemic theories	expansions
default theories	extensions
first-order programs	stable Herbrand models

Potassco

ASP-style playing with blocks

Logic program

```
on(a,b).
on(b,c).

above(X,Y) :- on(X,Y).
above(X,Y) :- on(X,Z), above(Z,Y).
```

Stable Herbrand model

$\{ \texttt{on(a,b)}, \texttt{on(b,c)}, \texttt{above(b,c)}, \texttt{above(a,b)}, \texttt{above(a,c)} \}$

Potassco

ASP-style playing with blocks

Logic program

```
on(a,b).
on(b,c).

above(X,Y) :- on(X,Y).
above(X,Y) :- on(X,Z), above(Z,Y).
```

Stable Herbrand model

{ on(a, b), on(b, c), above(b, c), above(a, b), above(a, c) }

Potassco

ASP-style playing with blocks

Logic program

```
on(a,b).
on(b,c).

above(X,Y) :- on(X,Y).
above(X,Y) :- on(X,Z), above(Z,Y).
```

Stable Herbrand model (and no others)

$\{ \texttt{on(a,b)}, \texttt{on(b,c)}, \texttt{above(b,c)}, \texttt{above(a,b)}, \texttt{above(a,c)} \}$

Potassco

ASP-style playing with blocks

Logic program

```
on(a,b).
on(b,c).

above(X,Y) :- above(Z,Y), on(X,Z).
above(X,Y) :- on(X,Y).
```

Stable Herbrand model (and no others)

{ on(a, b), on(b, c), above(b, c), above(a, b), above(a, c) }

ASP versus LP

ASP	Prolog
Model generation	Query orientation
Bottom-up	Top-down
Modeling language	Programming language
Rule-based format	
Instantiation Flat terms	Unification Nested terms
(Turing +) $NP(^{NP})$	Turing

ASP versus SAT

ASP	SAT
Model generation	
Bottom-up	
Constructive Logic	Classical Logic
Closed (and open) world reasoning	Open world reasoning
Modeling language	—
Complex reasoning modes	Satisfiability testing
Satisfiability	Satisfiability
Enumeration/Projection	—
Intersection/Union	—
Optimization	—
(Turing +) $NP(^{NP})$	NP

Outline

Potassco

Propositional Normal Logic Programs

- A logic program P is a set of rules of the form

$$\underbrace{a}_{\text{head}} \leftarrow \underbrace{b_1, \ldots, b_m, \neg c_1, \ldots, \neg c_n}_{\text{body}}$$

  - a and all b_i, c_j are atoms (propositional variables)
  - $\leftarrow$, ,, $\neg$ denote if, and, and negation
  - intuitive reading: head must be true if body holds

- Semantics given by stable models, informally,
  models of P justifying each true atom by some rule in P

Logic Programs

- A logic program P is a set of rules of the form

$$\underbrace{a}_{\text{head}} \leftarrow \underbrace{b_1, \ldots, b_m, \neg c_1, \ldots, \neg c_n}_{\text{body}}$$

  - a and all b_i, c_j are atoms (propositional variables)
  - $\leftarrow$, ,, $\neg$ denote if, and, and negation
  - intuitive reading: head must be true if body holds

- Semantics given by stable models, informally,
  models of P justifying each true atom by some rule in P

Logic Programs

- A logic program P is a set of rules of the form

$$\underbrace{a}_{\text{head}} \leftarrow \underbrace{b_1, \ldots, b_m, \neg c_1, \ldots, \neg c_n}_{\text{body}}$$

  - a and all b_i, c_j are atoms (propositional variables)
  - $\leftarrow$, ,, $\neg$ denote if, and, and negation
  - intuitive reading: head must be true if body holds

- Semantics given by stable models, informally,
  models of P justifying each true atom by some rule in P

Normal Logic Programs

- A logic program P is a set of rules of the form

$$\underbrace{a}_{\text{head}} \leftarrow \underbrace{b_1, \ldots, b_m, \neg c_1, \ldots, \neg c_n}_{\text{body}}$$

  - a and all b_i, c_j are atoms (propositional variables)
  - $\leftarrow$, ,, $\neg$ denote if, and, and negation
  - intuitive reading: head must be true if body holds

- Semantics given by stable models, informally,
  models of P justifying each true atom by some rule in P

- Disclaimer The following formalities apply to normal logic programs

Some truth tabling, back to SAT

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$
F	F	F	
F	F	T	
F	T	F	
F	T	T	
T	F	F	
T	F	T	
T	T	F	
T	T	T	

Some truth tabling, back to SAT

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$
F	**F**	**F**	$(\neg\mathbf{F} \rightarrow \mathbf{F}) \wedge (\mathbf{F} \rightarrow \mathbf{F})$
F	**F**	**T**	$(\neg\mathbf{F} \rightarrow \mathbf{F}) \wedge (\mathbf{F} \rightarrow \mathbf{T})$
F	**T**	**F**	$(\neg\mathbf{T} \rightarrow \mathbf{F}) \wedge (\mathbf{T} \rightarrow \mathbf{F})$
F	**T**	**T**	$(\neg\mathbf{T} \rightarrow \mathbf{F}) \wedge (\mathbf{T} \rightarrow \mathbf{T})$
T	**F**	**F**	$(\neg\mathbf{F} \rightarrow \mathbf{T}) \wedge (\mathbf{F} \rightarrow \mathbf{F})$
T	**F**	**T**	$(\neg\mathbf{F} \rightarrow \mathbf{T}) \wedge (\mathbf{F} \rightarrow \mathbf{T})$
T	**T**	**F**	$(\neg\mathbf{T} \rightarrow \mathbf{T}) \wedge (\mathbf{T} \rightarrow \mathbf{F})$
T	**T**	**T**	$(\neg\mathbf{T} \rightarrow \mathbf{T}) \wedge (\mathbf{T} \rightarrow \mathbf{T})$

Potassco

Some truth tabling, back to SAT

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$
F	**F**	**F**	$(\mathbf{T} \rightarrow \mathbf{F}) \wedge (\mathbf{F} \rightarrow \mathbf{F})$
F	**F**	**T**	$(\mathbf{T} \rightarrow \mathbf{F}) \wedge (\mathbf{F} \rightarrow \mathbf{T})$
F	**T**	**F**	$(\mathbf{F} \rightarrow \mathbf{F}) \wedge (\mathbf{T} \rightarrow \mathbf{F})$
F	**T**	**T**	$(\mathbf{F} \rightarrow \mathbf{F}) \wedge (\mathbf{T} \rightarrow \mathbf{T})$
T	**F**	**F**	$(\mathbf{T} \rightarrow \mathbf{T}) \wedge (\mathbf{F} \rightarrow \mathbf{F})$
T	**F**	**T**	$(\mathbf{T} \rightarrow \mathbf{T}) \wedge (\mathbf{F} \rightarrow \mathbf{T})$
T	**T**	**F**	$(\mathbf{F} \rightarrow \mathbf{T}) \wedge (\mathbf{T} \rightarrow \mathbf{F})$
T	**T**	**T**	$(\mathbf{F} \rightarrow \mathbf{T}) \wedge (\mathbf{T} \rightarrow \mathbf{T})$

Potassco

Some truth tabling, back to SAT

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$
F	F	F	$(\mathbf{T} \rightarrow \mathbf{F}) \wedge (\mathbf{F} \rightarrow \mathbf{F})$
F	F	T	$(\mathbf{T} \rightarrow \mathbf{F}) \wedge (\mathbf{F} \rightarrow \mathbf{T})$
F	T	F	$(\mathbf{F} \rightarrow \mathbf{F}) \wedge (\mathbf{T} \rightarrow \mathbf{F})$
F	T	T	$(\mathbf{F} \rightarrow \mathbf{F}) \wedge (\mathbf{T} \rightarrow \mathbf{T})$
T	F	F	$(\mathbf{T} \rightarrow \mathbf{T}) \wedge (\mathbf{F} \rightarrow \mathbf{F})$
T	F	T	$(\mathbf{T} \rightarrow \mathbf{T}) \wedge (\mathbf{F} \rightarrow \mathbf{T})$
T	T	F	$(\mathbf{F} \rightarrow \mathbf{T}) \wedge (\mathbf{T} \rightarrow \mathbf{F})$
T	T	T	$(\mathbf{F} \rightarrow \mathbf{T}) \wedge (\mathbf{T} \rightarrow \mathbf{T})$

Potassco

Some truth tabling, back to SAT

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$
F	**F**	**F**	$\mathbf{F} \wedge (\mathbf{F} \rightarrow \mathbf{F})$
F	**F**	**T**	$\mathbf{F} \wedge (\mathbf{F} \rightarrow \mathbf{T})$
F	**T**	**F**	$(\mathbf{F} \rightarrow \mathbf{F}) \wedge \mathbf{F}$
F	**T**	**T**	$(\mathbf{F} \rightarrow \mathbf{F}) \wedge (\mathbf{T} \rightarrow \mathbf{T})$
T	**F**	**F**	$(\mathbf{T} \rightarrow \mathbf{T}) \wedge (\mathbf{F} \rightarrow \mathbf{F})$
T	**F**	**T**	$(\mathbf{T} \rightarrow \mathbf{T}) \wedge (\mathbf{F} \rightarrow \mathbf{T})$
T	**T**	**F**	$(\mathbf{F} \rightarrow \mathbf{T}) \wedge \mathbf{F}$
T	**T**	**T**	$(\mathbf{F} \rightarrow \mathbf{T}) \wedge (\mathbf{T} \rightarrow \mathbf{T})$

Potassco

Some truth tabling, back to SAT

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$
F	F	F	$\mathbf{F} \wedge (\mathbf{F} \rightarrow \mathbf{F})$
F	F	T	$\mathbf{F} \wedge (\mathbf{F} \rightarrow \mathbf{T})$
F	T	F	$(\mathbf{F} \rightarrow \mathbf{F}) \wedge \mathbf{F}$
F	T	T	$(\mathbf{F} \rightarrow \mathbf{F}) \wedge (\mathbf{T} \rightarrow \mathbf{T})$
T	F	F	$(\mathbf{T} \rightarrow \mathbf{T}) \wedge (\mathbf{F} \rightarrow \mathbf{F})$
T	F	T	$(\mathbf{T} \rightarrow \mathbf{T}) \wedge (\mathbf{F} \rightarrow \mathbf{T})$
T	T	F	$(\mathbf{F} \rightarrow \mathbf{T}) \wedge \mathbf{F}$
T	T	T	$(\mathbf{F} \rightarrow \mathbf{T}) \wedge (\mathbf{T} \rightarrow \mathbf{T})$

Potassco

Some truth tabling, back to SAT

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$
F	F	F	F $\wedge$ T
F	F	T	F $\wedge$ T
F	T	F	T $\wedge$ F
F	T	T	T $\wedge$ T
T	F	F	T $\wedge$ T
T	F	T	T $\wedge$ T
T	T	F	T $\wedge$ F
T	T	T	T $\wedge$ T

Potassco

Some truth tabling, back to SAT

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$
F	F	F	F
F	F	T	F
F	T	F	F
F	T	T	T
T	F	F	T
T	F	T	T
T	T	F	F
T	T	T	T

Some truth tabling, back to SAT

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$
F	F	F	F
F	F	T	F
F	T	F	F
F	T	T	T
T	F	F	T
T	F	T	T
T	T	F	F
T	T	T	T

- We get four models: $\{b, c\}$, $\{a\}$, $\{a, c\}$, and $\{a, b, c\}$

Some truth tabling, and now ASP

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$
F	F	F	
F	F	T	
F	T	F	
F	T	T	
T	F	F	
T	F	T	
T	T	F	
T	T	T	

Potassco

Some truth tabling, and now ASP

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$
F	**F**	**F**	$(\neg \mathbf{F} \rightarrow a) \wedge (b \rightarrow c)$
F	**F**	**T**	$(\neg \mathbf{F} \rightarrow a) \wedge (b \rightarrow c)$
F	**T**	**F**	$(\neg \mathbf{T} \rightarrow a) \wedge (b \rightarrow c)$
F	**T**	**T**	$(\neg \mathbf{T} \rightarrow a) \wedge (b \rightarrow c)$
T	**F**	**F**	$(\neg \mathbf{F} \rightarrow a) \wedge (b \rightarrow c)$
T	**F**	**T**	$(\neg \mathbf{F} \rightarrow a) \wedge (b \rightarrow c)$
T	**T**	**F**	$(\neg \mathbf{T} \rightarrow a) \wedge (b \rightarrow c)$
T	**T**	**T**	$(\neg \mathbf{T} \rightarrow a) \wedge (b \rightarrow c)$

Potassco

Some truth tabling, and now ASP

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$
F	**F**	**F**	$(\mathbf{T} \rightarrow a) \wedge (b \rightarrow c)$
F	**F**	**T**	$(\mathbf{T} \rightarrow a) \wedge (b \rightarrow c)$
F	**T**	**F**	$(\mathbf{F} \rightarrow a) \wedge (b \rightarrow c)$
F	**T**	**T**	$(\mathbf{F} \rightarrow a) \wedge (b \rightarrow c)$
T	**F**	**F**	$(\mathbf{T} \rightarrow a) \wedge (b \rightarrow c)$
T	**F**	**T**	$(\mathbf{T} \rightarrow a) \wedge (b \rightarrow c)$
T	**T**	**F**	$(\mathbf{F} \rightarrow a) \wedge (b \rightarrow c)$
T	**T**	**T**	$(\mathbf{F} \rightarrow a) \wedge (b \rightarrow c)$

Potassco

Some truth tabling, and now ASP

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$
F	**F**	**F**	$a \wedge (b \rightarrow c)$
F	**F**	**T**	$a \wedge (b \rightarrow c)$
F	**T**	**F**	$(\mathbf{F} \rightarrow a) \wedge (b \rightarrow c)$
F	**T**	**T**	$(\mathbf{F} \rightarrow a) \wedge (b \rightarrow c)$
T	**F**	**F**	$a \wedge (b \rightarrow c)$
T	**F**	**T**	$a \wedge (b \rightarrow c)$
T	**T**	**F**	$(\mathbf{F} \rightarrow a) \wedge (b \rightarrow c)$
T	**T**	**T**	$(\mathbf{F} \rightarrow a) \wedge (b \rightarrow c)$

Potassco

Some truth tabling, and now ASP

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$
F	**F**	**F**	$a \wedge (b \rightarrow c)$
F	**F**	**T**	$a \wedge (b \rightarrow c)$
F	**T**	**F**	$(\mathbf{F} \rightarrow a) \wedge (b \rightarrow c)$
F	**T**	**T**	$(\mathbf{F} \rightarrow a) \wedge (b \rightarrow c)$
T	**F**	**F**	$a \wedge (b \rightarrow c)$
T	**F**	**T**	$a \wedge (b \rightarrow c)$
T	**T**	**F**	$(\mathbf{F} \rightarrow a) \wedge (b \rightarrow c)$
T	**T**	**T**	$(\mathbf{F} \rightarrow a) \wedge (b \rightarrow c)$

Potassco

Some truth tabling, and now ASP

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$
F	**F**	**F**	$a \wedge (b \rightarrow c)$
F	**F**	**T**	$a \wedge (b \rightarrow c)$
F	**T**	**F**	$\mathbf{T} \wedge (b \rightarrow c)$
F	**T**	**T**	$\mathbf{T} \wedge (b \rightarrow c)$
T	**F**	**F**	$a \wedge (b \rightarrow c)$
T	**F**	**T**	$a \wedge (b \rightarrow c)$
T	**T**	**F**	$\mathbf{T} \wedge (b \rightarrow c)$
T	**T**	**T**	$\mathbf{T} \wedge (b \rightarrow c)$

Potassco

Some truth tabling, and now ASP

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$
F	**F**	**F**	$a \wedge (b \rightarrow c)$
F	**F**	**T**	$a \wedge (b \rightarrow c)$
F	**T**	**F**	$(b \rightarrow c)$
F	**T**	**T**	$(b \rightarrow c)$
T	**F**	**F**	$a \wedge (b \rightarrow c)$
T	**F**	**T**	$a \wedge (b \rightarrow c)$
T	**T**	**F**	$(b \rightarrow c)$
T	**T**	**T**	$(b \rightarrow c)$

Reduct

Some truth tabling, and now ASP

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$
F	F	F	$a \wedge (b \rightarrow c)$
F	F	T	$a \wedge (b \rightarrow c)$
F	T	F	$(b \rightarrow c)$
F	T	T	$(b \rightarrow c)$
T	F	F	$a \wedge (b \rightarrow c)$
T	F	T	$a \wedge (b \rightarrow c)$
T	T	F	$(b \rightarrow c)$
T	T	T	$(b \rightarrow c)$

Reduct

Potassco

Some truth tabling, and now ASP

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$	
F	F	F	$a \wedge (b \rightarrow c)$	
F	F	T	$a \wedge (b \rightarrow c)$	
F	T	F	$(b \rightarrow c)$	
F	T	T	$(b \rightarrow c)$	$\models$
T	F	F	$a \wedge (b \rightarrow c)$	$\models a$
T	F	T	$a \wedge (b \rightarrow c)$	$\models a$
T	T	F	$(b \rightarrow c)$	
T	T	T	$(b \rightarrow c)$	$\models$

Reduct

Some truth tabling, and now ASP

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$	
F	F	F	$a \wedge (b \rightarrow c)$	
F	F	T	$a \wedge (b \rightarrow c)$	
F	T	F	$(b \rightarrow c)$	
F	T	T	$(b \rightarrow c)$	$\models$
T	F	F	$a \wedge (b \rightarrow c)$	$\models a$
T	F	T	$a \wedge (b \rightarrow c)$	$\models a$
T	T	F	$(b \rightarrow c)$	
T	T	T	$(b \rightarrow c)$	$\models$

Reduct

Some truth tabling, and now ASP

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$	
F	F	F	$a \wedge (b \rightarrow c)$	$\models a$
F	F	T	$a \wedge (b \rightarrow c)$	$\models a$
F	T	F	$(b \rightarrow c)$	$\models$
F	T	T	$(b \rightarrow c)$	$\models$
T	F	F	$a \wedge (b \rightarrow c)$	$\models a$
T	F	T	$a \wedge (b \rightarrow c)$	$\models a$
T	T	F	$(b \rightarrow c)$	$\models$
T	T	T	$(b \rightarrow c)$	$\models$

Reduct

Potassco

Some truth tabling, and now ASP

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$	
F	**F**	**F**	$a \wedge (b \rightarrow c)$	
F	**F**	**T**	$a \wedge (b \rightarrow c)$	
F	**T**	**F**	$(b \rightarrow c)$	
F	**T**	**T**	$(b \rightarrow c)$	
T	**F**	**F**	$a \wedge (b \rightarrow c)$	$\models a$ Stable model
T	**F**	**T**	$a \wedge (b \rightarrow c)$	
T	**T**	**F**	$(b \rightarrow c)$	
T	**T**	**T**	$(b \rightarrow c)$	

Reduct

- We get one stable model: $\{a\}$

Some truth tabling, and now ASP

a	b	c	$(\neg b \rightarrow a) \wedge (b \rightarrow c)$	
F	F	F	$a \wedge (b \rightarrow c)$	
F	F	T	$a \wedge (b \rightarrow c)$	
F	T	F	$(b \rightarrow c)$	
F	T	T	$(b \rightarrow c)$	
T	F	F	$a \wedge (b \rightarrow c)$	$\models a$ Stable model
T	F	T	$a \wedge (b \rightarrow c)$	
T	T	F	$(b \rightarrow c)$	
T	T	T	$(b \rightarrow c)$	
			Reduct	

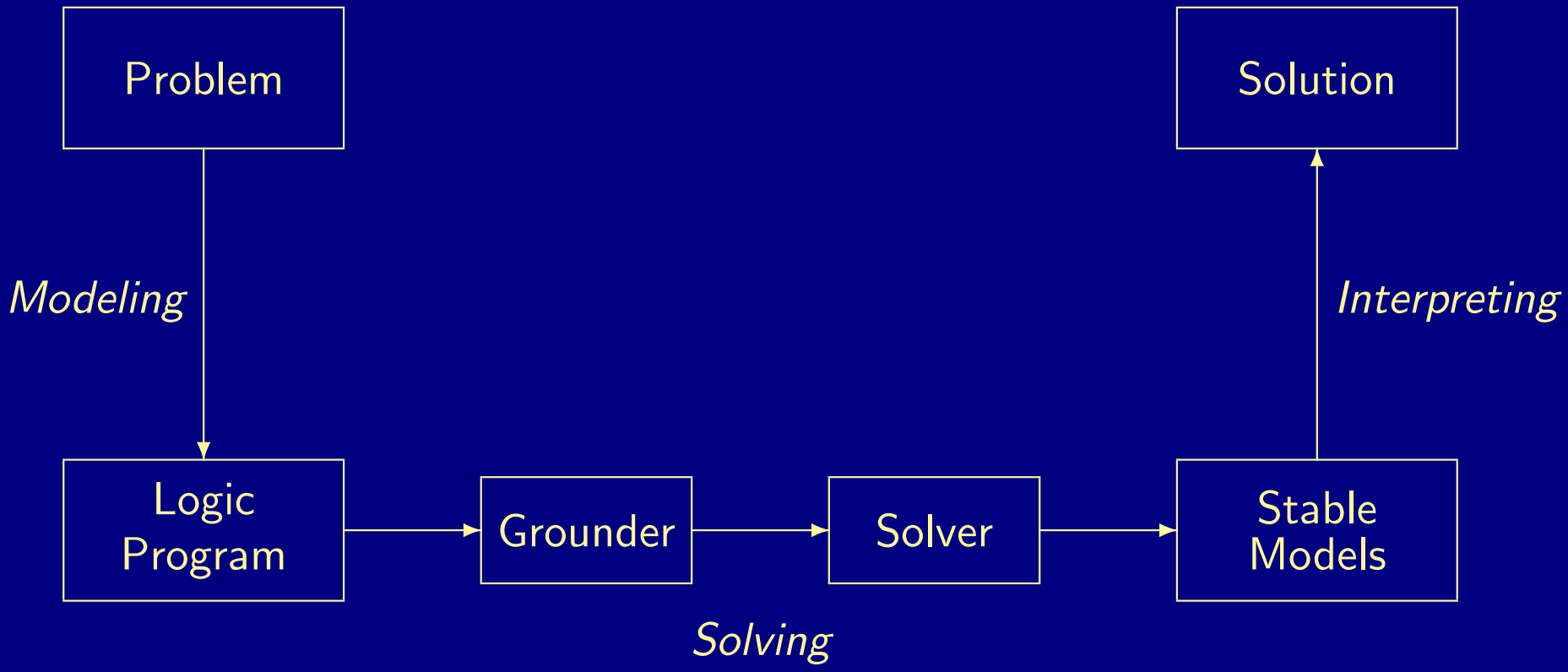
- We get one stable model: $\{a\}$
- Stable models = Smallest models of (respective) reducts

Potassco

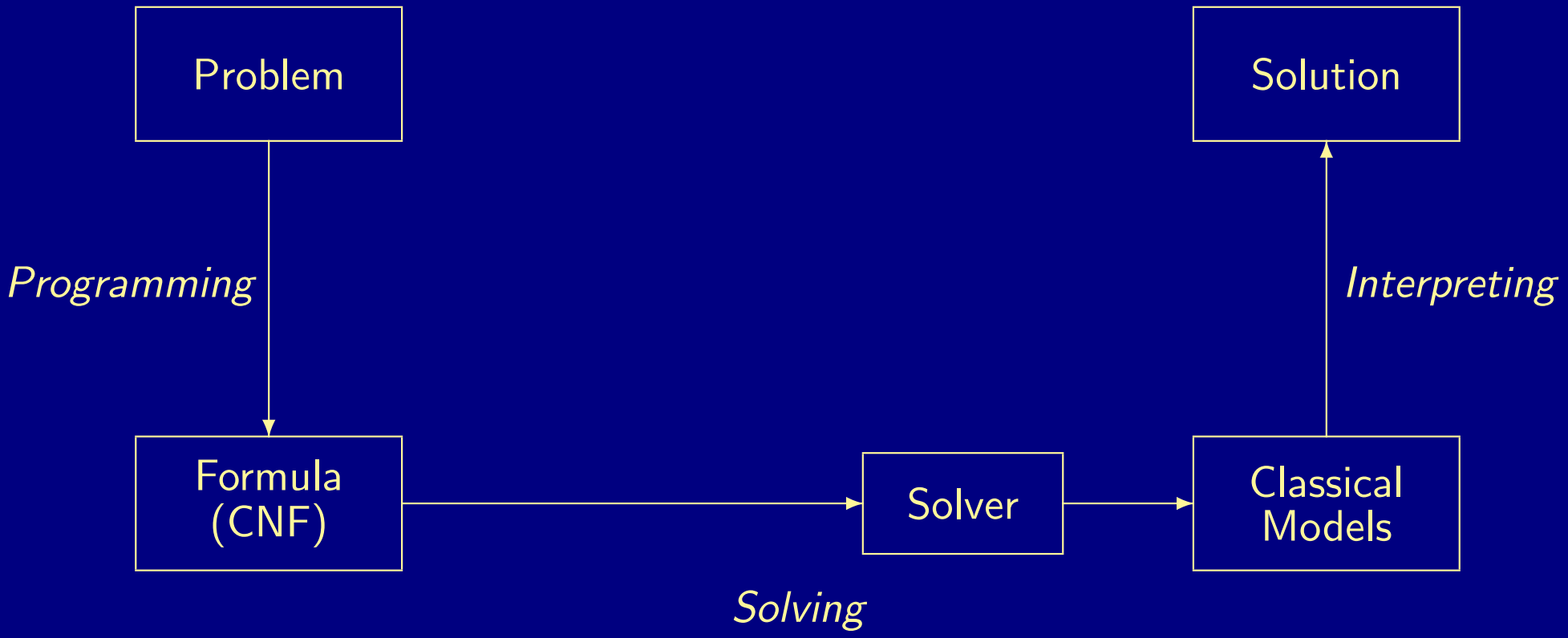
Outline

Potassco

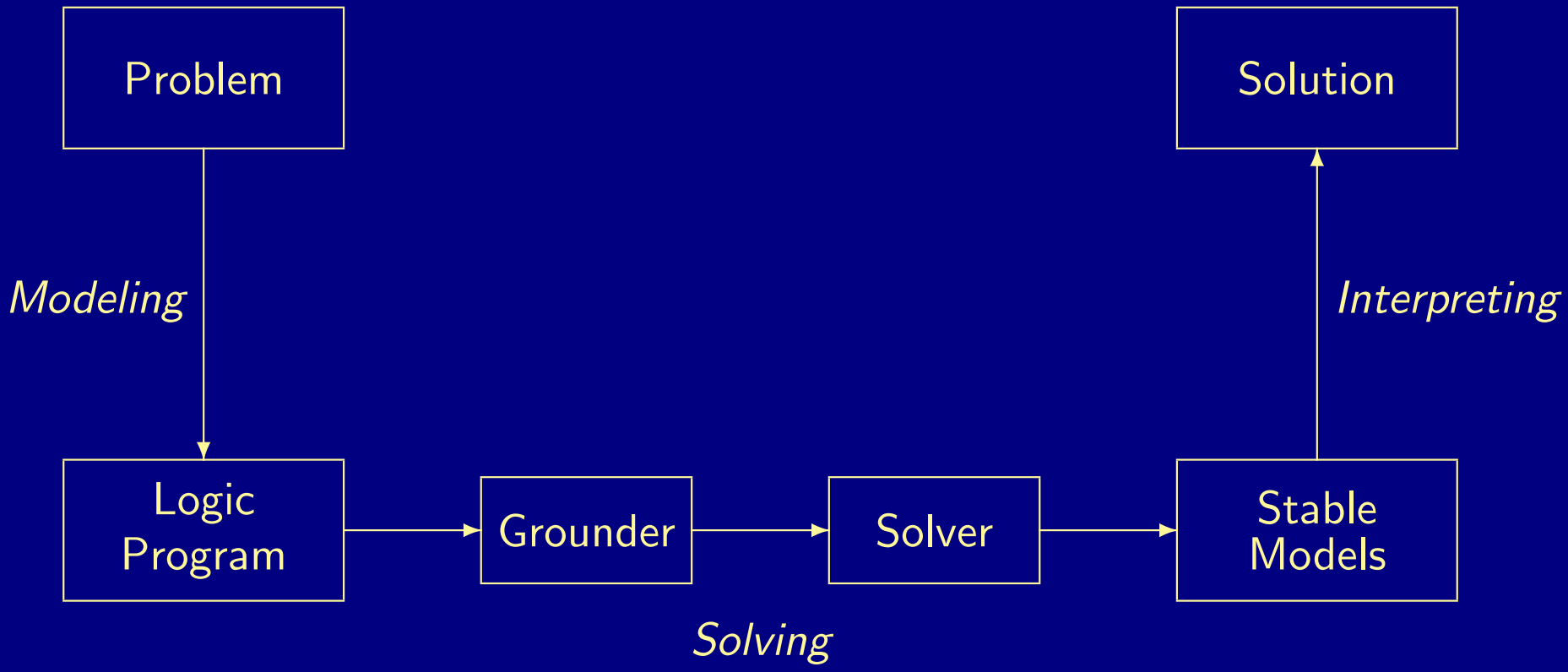
ASP modeling, grounding, and solving



SAT solving

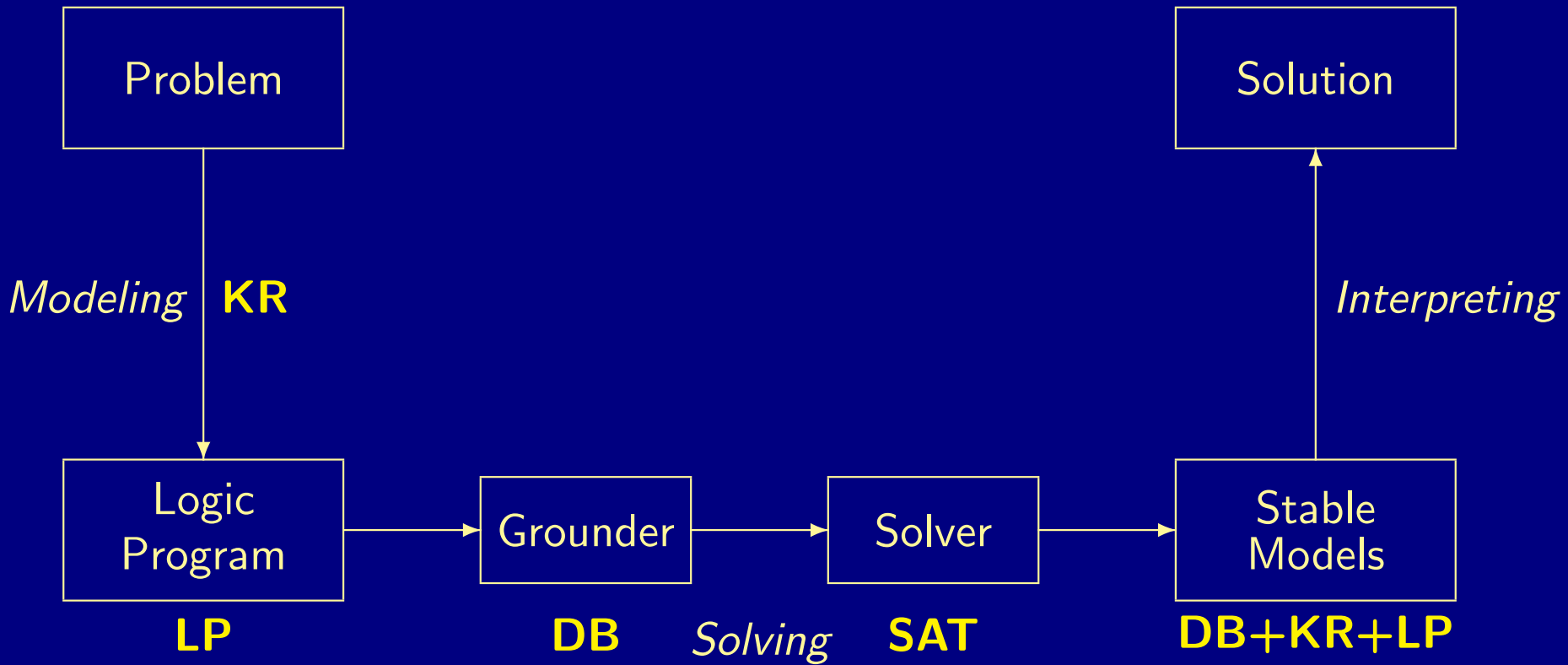

Problem
Solution
Programming
Interpreting
Formula
(CNF)
Solver
Classical
Models
Solving

Rooting ASP solving



Potassco

Rooting ASP solving

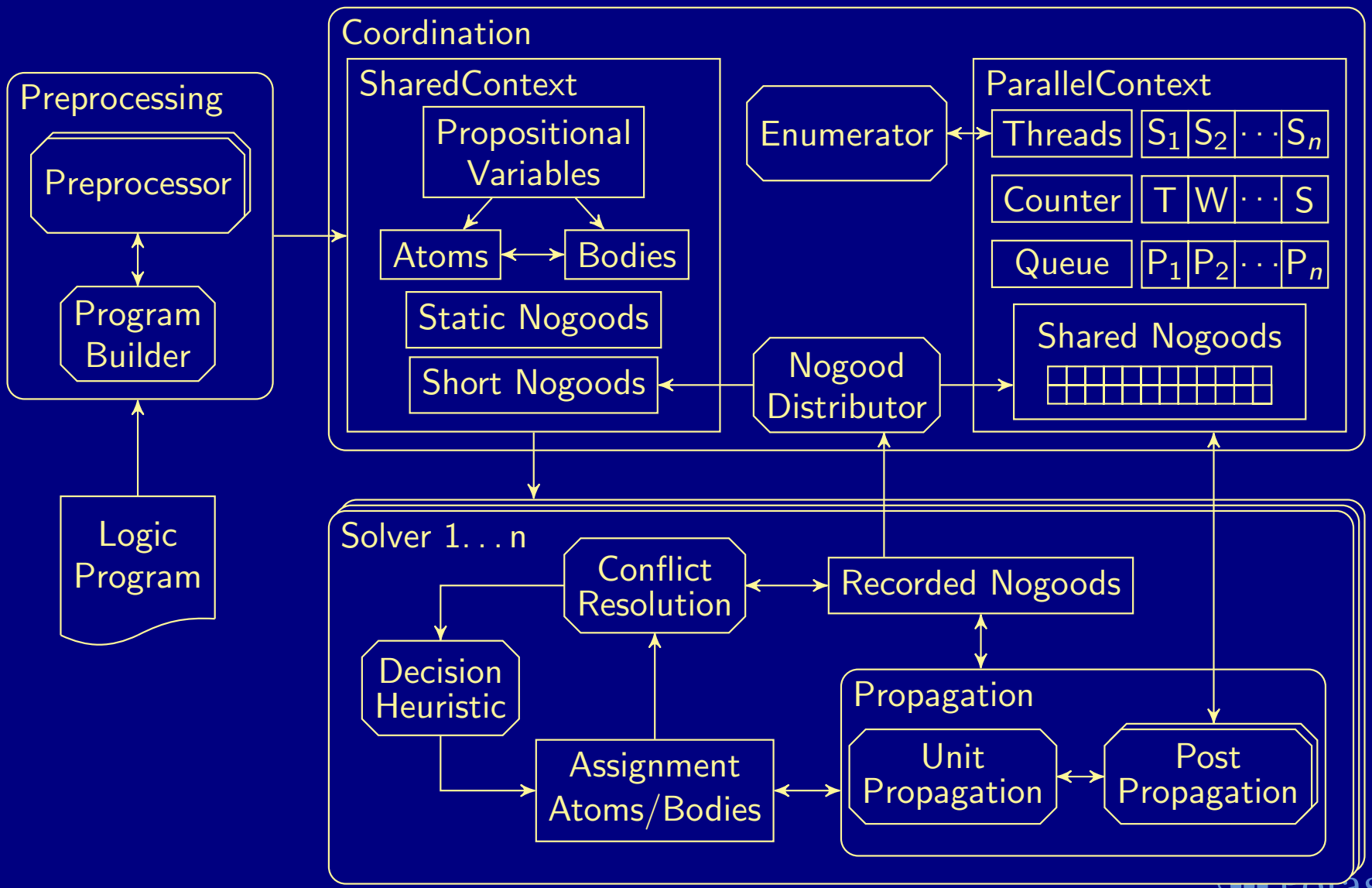


Potassco

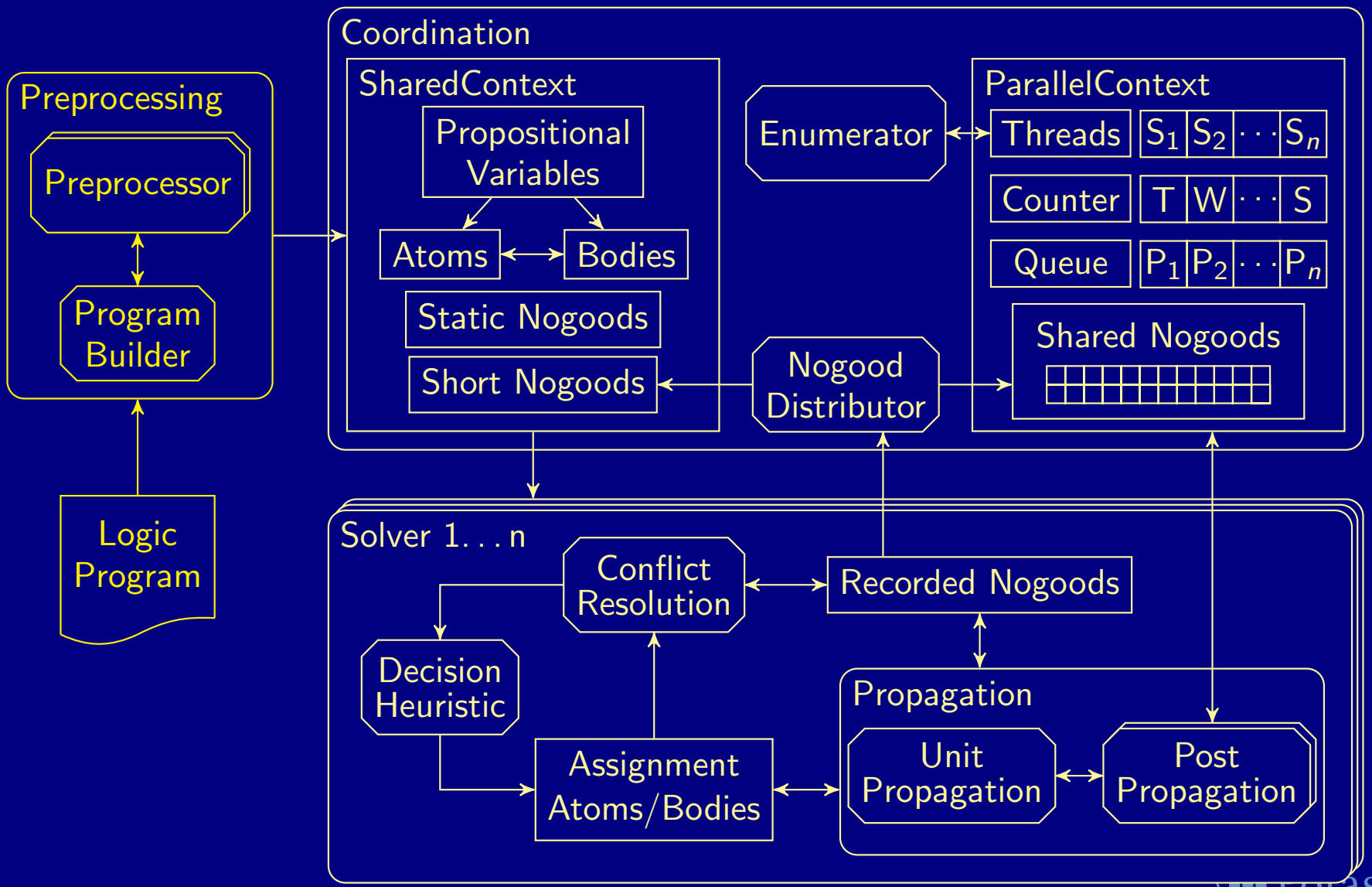
Outline

Potassco

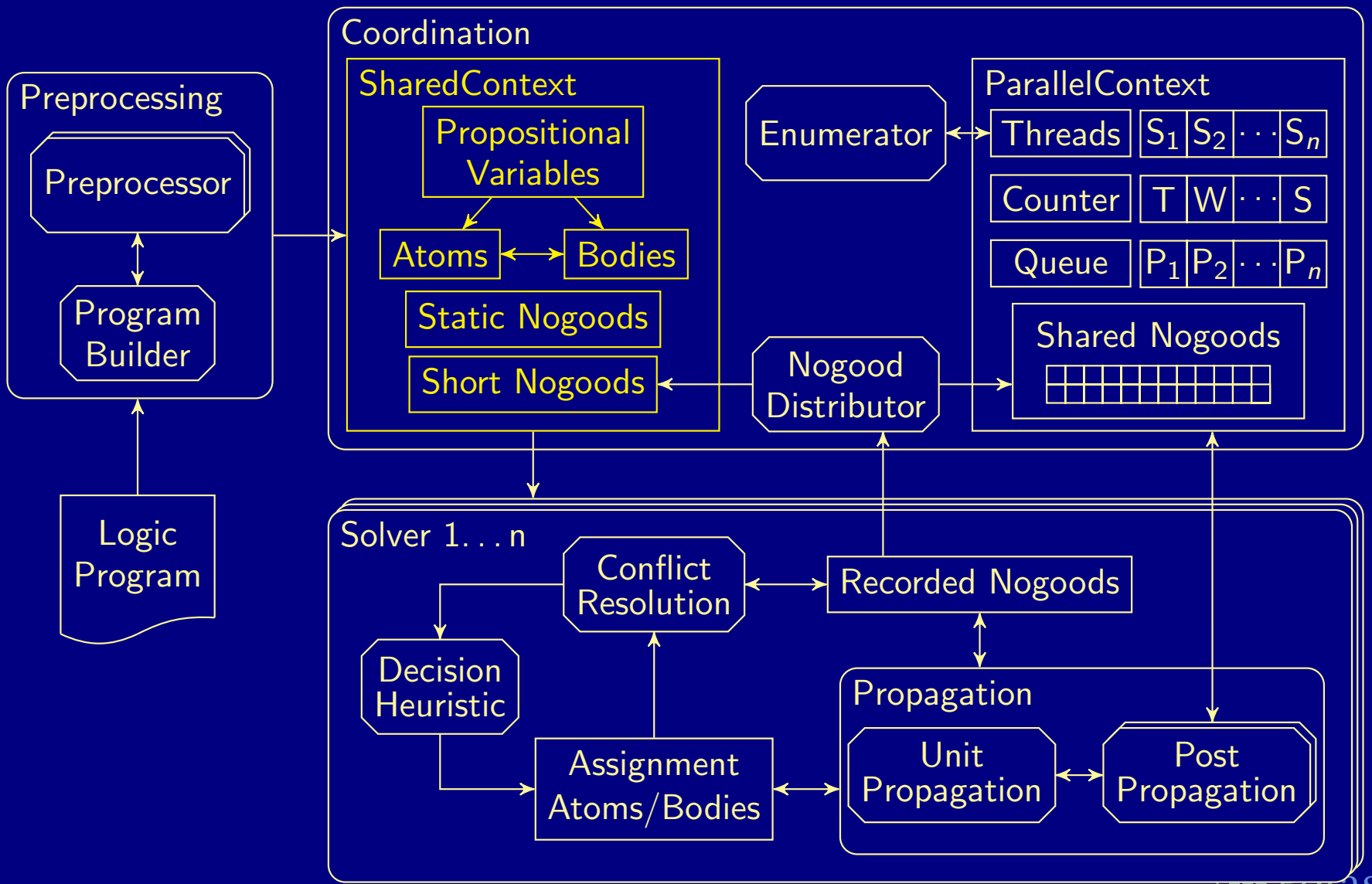
Multi-threaded architecture of *clasp*



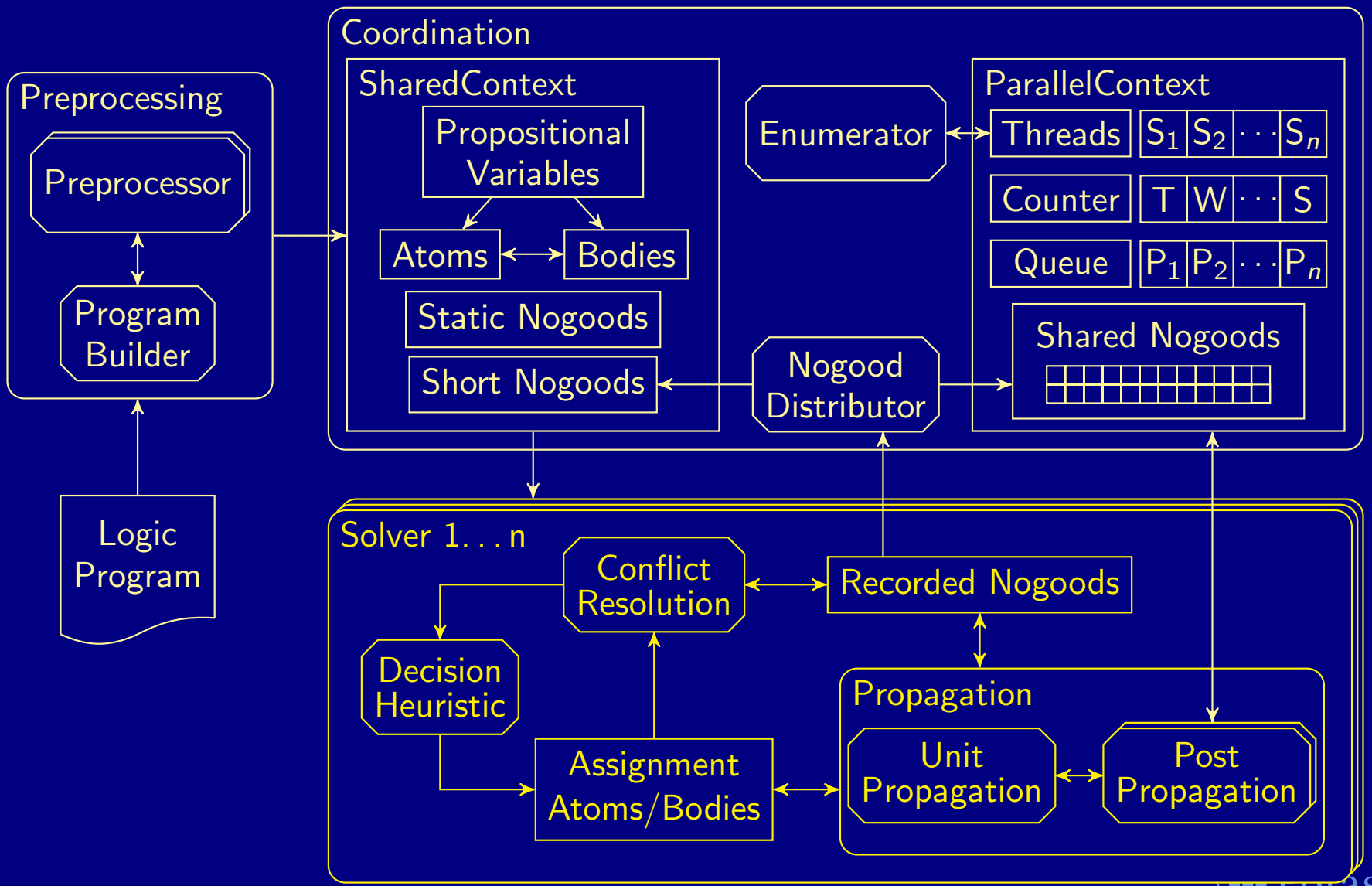
Multi-threaded architecture of *clasp*



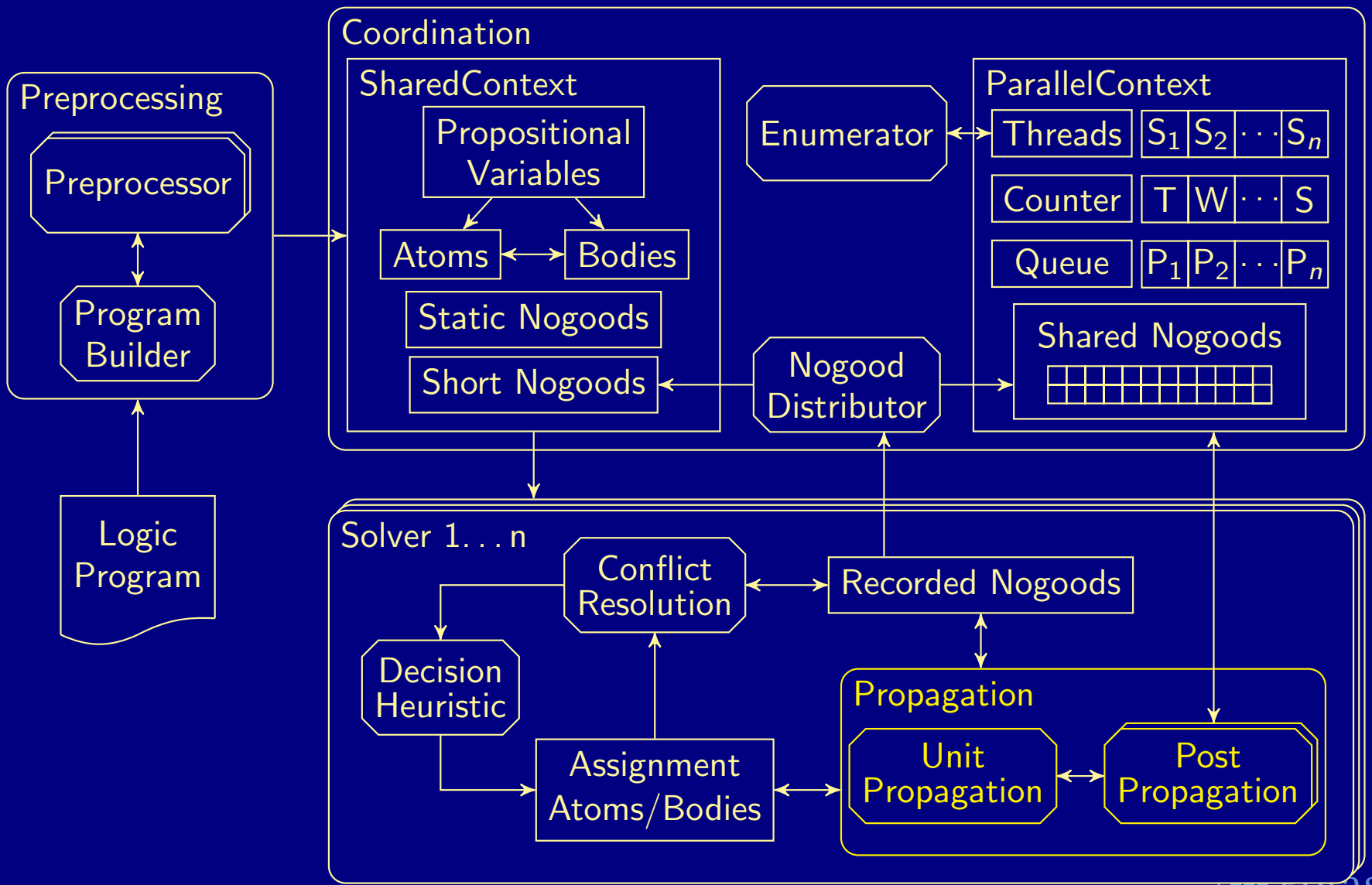
Multi-threaded architecture of *clasp*



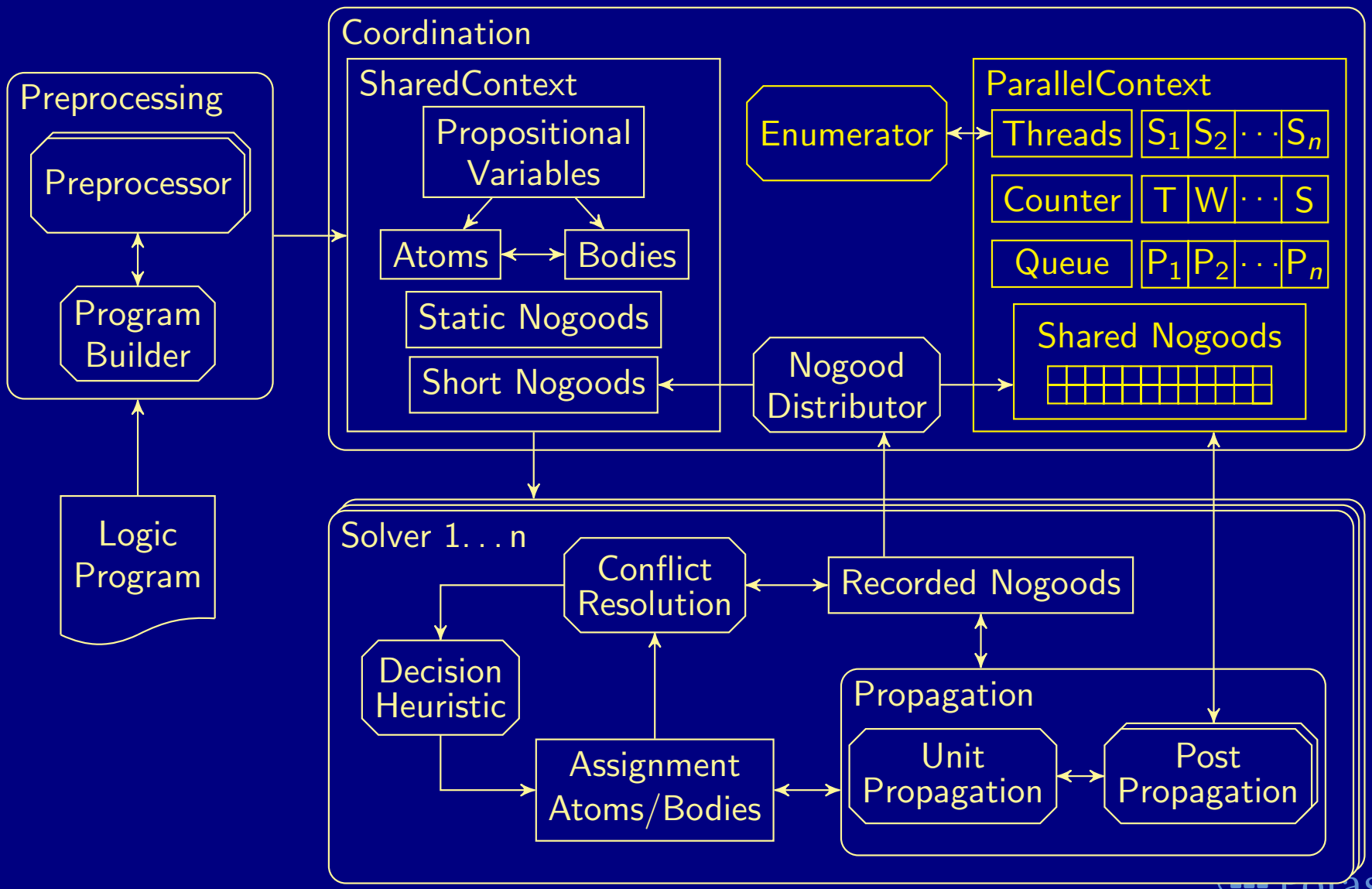
Multi-threaded architecture of *clasp*



Multi-threaded architecture of *clasp*



Multi-threaded architecture of *clasp*



Outline

Potassco

Two sides of a coin

- ASP as High-level Language
  - Express problem instance as sets of facts
  - Encode problem class as a set of rules
  - Read off solutions from stable models of facts and rules
- ASP as Low-level Language
  - Compile a problem into a set of facts and rules
  - Solve the original problem by solving its compilation
- ASP and Imperative language

  Control continuously changing logic programs

Potassco

Two sides of a coin

- ASP as High-level Language
  - Express problem instance as sets of facts
  - Encode problem class as a set of rules
  - Read off solutions from stable models of facts and rules
- ASP as Low-level Language
  - Compile a problem into a set of facts and rules
  - Solve the original problem by solving its compilation
- ASP and Imperative language

  Control continuously changing logic programs

Two sides of a coin

- ASP as High-level Language
  - Express problem instance as sets of facts
  - Encode problem class as a set of rules
  - Read off solutions from stable models of facts and rules
- ASP as Low-level Language
  - Compile a problem into a set of facts and rules
  - Solve the original problem by solving its compilation
- ASP and Imperative language
  - Control continuously changing logic programs

Potassco

Two sides of a coin

- ASP as High-level Language
  - Express problem instance as sets of facts
  - Encode problem class as a set of rules
  - Read off solutions from stable models of facts and rules
- ASP as "Low-level" Language
  - Compile a problem instance into a set of facts
  - Encode problem class as a set of rules
  - Solve the original problem by solving its compilation
- ASP and Imperative language
  - Control continuously changing logic programs

Potassco

Two and a half sides of a coin

- ASP as High-level Language
  - Express problem instance as sets of facts
  - Encode problem class as a set of rules
  - Read off solutions from stable models of facts and rules
- ASP as "Low-level" Language
  - Compile a problem instance into a set of facts
  - Encode problem class as a set of rules
  - Solve the original problem by solving its compilation
- ASP and Imperative language
  - Control continuously changing logic programs

Potassco

Outline

Potassco

Upcoming experience

- ASP is a viable tool for Knowledge Representation and Reasoning
  - Integration of DB, LP, KR, and SAT techniques
  - Combinatorial search problems in the realm of NP and NP^{NP}
  - Succinct, elaboration-tolerant problem representations
    - rapid application development tool
  - Easy handling of knowledge-intensive applications
    - data, defaults, exceptions, frame axioms, reachability etc
- ASP offers efficient and versatile off-the-shelf solving technology
  - `http://potassco.org`
  - winning ASP, CASC, MISC, PB, and SAT competitions
- ASP has a growing range of applications, and its's good fun!

Potassco

Upcoming experience

- ASP is a viable tool for Knowledge Representation and Reasoning
  - Integration of DB, LP, KR, and SAT techniques
  - Combinatorial search problems in the realm of NP and NP^{NP}
  - Succinct, elaboration-tolerant problem representations
    - rapid application development tool
  - Easy handling of knowledge-intensive applications
    - data, defaults, exceptions, frame axioms, reachability etc
- ASP offers efficient and versatile off-the-shelf solving technology
  - `http://potassco.org`
  - winning ASP, CASC, MISC, PB, and SAT competitions
- ASP has a growing range of applications, and its's good fun!

ASP = DB+LP+KR+SAT

Potassco

Upcoming experience

- ASP is a viable tool for Knowledge Representation and Reasoning
  - Integration of DB, LP, KR, and SAT techniques
  - Combinatorial search problems in the realm of NP and NP^{NP}
  - Succinct, elaboration-tolerant problem representations
    - rapid application development tool
  - Easy handling of knowledge-intensive applications
    - data, defaults, exceptions, frame axioms, reachability etc
- ASP offers efficient and versatile off-the-shelf solving technology
  - `http://potassco.org`
  - winning ASP, CASC, MISC, PB, and SAT competitions
- ASP has a growing range of applications, and its's good fun!

ASP = DB+LP+KR+SMT^n

Potassco

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

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

[3] 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.

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

[5] S. Baselice, P. Bonatti, and M. Gelfond.
Towards an integration of answer set and constraint solving

Potassco

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.

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

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

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

Potassco

IOS Press, 2009.

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

[10] G. Brewka, I. Niemelä, and M. Truszczyński.
Answer set optimization.
In G. Gottlob and T. Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 867–872. Morgan Kaufmann Publishers, 2003.

[11] K. Clark.
Negation as failure.
In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, 1978.

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

Potassco

[13] 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.

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

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

[16] E. Di Rosa, E. Giunchiglia, and M. Maratea.
Solving satisfiability problems with preferences.
Constraints, 15(4):485–515, 2010.

[17] C. Drescher, M. Gebser, T. Grote, B. Kaufmann, A. König, M. Ostrowski, and T. Schaub.

Potassco

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.

Potassco

[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.
Answer sets for propositional theories.

Potassco

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, A. Harrison, R. Kaminski, V. Lifschitz, and T. Schaub.
Abstract Gringo.
Theory and Practice of Logic Programming, 15(4-5):449–463, 2015.
Available at `http://arxiv.org/abs/1507.06576`.

Potassco

[27] M. Gebser, R. Kaminski, B. Kaufmann, M. Lindauer, M. Ostrowski, J. Romero, T. Schaub, and S. Thiele.
Potassco User Guide.
University of Potsdam, second edition edition, 2015.

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

[29] 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.

[30] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub.

Potassco

On the implementation of weight constraint rules in conflict-driven ASP solvers.
In Hill and Warren [49], pages 250–264.

[31] 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.

[32] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub.
clasp: A conflict-driven answer set solver.
In Baral et al. [3], pages 260–265.

[33] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub.
Conflict-driven answer set enumeration.
In Baral et al. [3], pages 136–148.

[34] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub.
Conflict-driven answer set solving.
In Veloso [74], pages 386–392.

Potassco

[35] 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.

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

[37] 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*

Potassco

(CPAIOR'09), volume 5547 of *Lecture Notes in Computer Science*, pages 71–86. Springer-Verlag, 2009.

[38] M. Gebser, M. Ostrowski, and T. Schaub.
Constraint answer set solving.
In Hill and Warren [49], pages 235–249.

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

[40] 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.

Potassco

[41] 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.

[42] M. Gelfond and Y. Kahl.
Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach.
Cambridge University Press, 2014.

[43] M. Gelfond and N. Leone.
Logic programming and knowledge representation — the A-Prolog perspective.
Artificial Intelligence, 138(1-2):3–38, 2002.

[44] M. Gelfond and V. Lifschitz.
The stable model semantics for logic programming.

Potassco

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.

[45] 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.

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

[47] K. Gödel.
Zum intuitionistischen Aussagenkalkül.
Anzeiger der Akademie der Wissenschaften in Wien, page 65–66, 1932.

[48] A. Heyting.

Potassco

Die formalen Regeln der intuitionistischen Logik.
In *Sitzungsberichte der Preussischen Akademie der Wissenschaften*, page 42–56. Deutsche Akademie der Wissenschaften zu Berlin, 1930. Reprint in Logik-Texte: Kommentierte Auswahl zur Geschichte der Modernen Logik, Akademie-Verlag, 1986.

[49] 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.

[50] J. Huang.
The effect of restarts on the efficiency of clause learning.
In Veloso [74], pages 2318–2323.

[51] 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.

[52] J. Lee.

Potassco

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.

[53] 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.

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

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

[56] V. Lifschitz and A. Razborov.
Why are there so many loop formulas?

Potassco

ACM Transactions on Computational Logic, 7(2):261–268, 2006.

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

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

[59] 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.

[60] J. Marques-Silva, I. Lynce, and S. Malik.
Conflict-driven clause learning SAT solvers.
In Biere et al. [8], chapter 4, pages 131–153.

[61] J. Marques-Silva and K. Sakallah.

Potassco

GRASP: A search algorithm for propositional satisfiability.
IEEE Transactions on Computers, 48(5):506–521, 1999.

[62] 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.

[63] 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.

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

Potassco

[65] 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.

[66] 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.

[67] 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.

[68] K. Pipatsrisawat and A. Darwiche.
A lightweight component caching scheme for satisfiability solvers.

Potassco

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.

[69] L. Ryan.
Efficient algorithms for clause-learning SAT solvers.
Master's thesis, Simon Fraser University, 2004.

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

[71] T. Son and E. Pontelli.
Planning with preferences using logic programming.
Theory and Practice of Logic Programming, 6(5):559–608, 2006.

[72] T. Syrjänen.
Lparse 1.0 user's manual, 2001.

[73] A. Van Gelder, K. Ross, and J. Schlipf.

Potassco

The well-founded semantics for general logic programs.
Journal of the ACM, 38(3):620–650, 1991.

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

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

Potassco