DE LA RECHERCHE À L'INDUSTRIE

SOPRANO : SMT MEETS CP

ProofInUse KickOff | F.Bobot

2 February

www.cea.fr

SOPRANO

- Partners: CEA, Univ. Paris-Sud, Inria Rennes, OcamlPro, Adacore
- 42 months

Design next generation of provers for program analysis (WP)

$\Longrightarrow$ Sat Modulo Theories & Constraint Programming

`Popop` one of the tools developped during the project.

SOPRANO

- Partners: CEA, Univ. Paris-Sud, Inria Rennes, OcamlPro, Adacore
- 42 months

Design next generation of provers for program analysis (WP)

$\Longrightarrow$ Sat Modulo Theories & Constraint Programming

`Popop` one of the tools developped during the project.

SOPRANO

- Partners: CEA, Univ. Paris-Sud, Inria Rennes, OcamlPro, Adacore
- 42 months

Design next generation of provers for program analysis (WP)

$\Longrightarrow$ Sat Modulo Theories & Constraint Programming

`Popop` one of the tools developped during the project.

SOPRANO

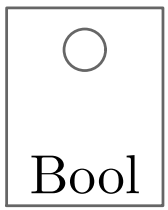
- Partners: CEA, Univ. Paris-Sud, Inria Rennes, OcamlPro, Adacore
- 42 months

Design next generation of provers for program analysis (WP)

$\Longrightarrow$ Sat Modulo Theories & Constraint Programming

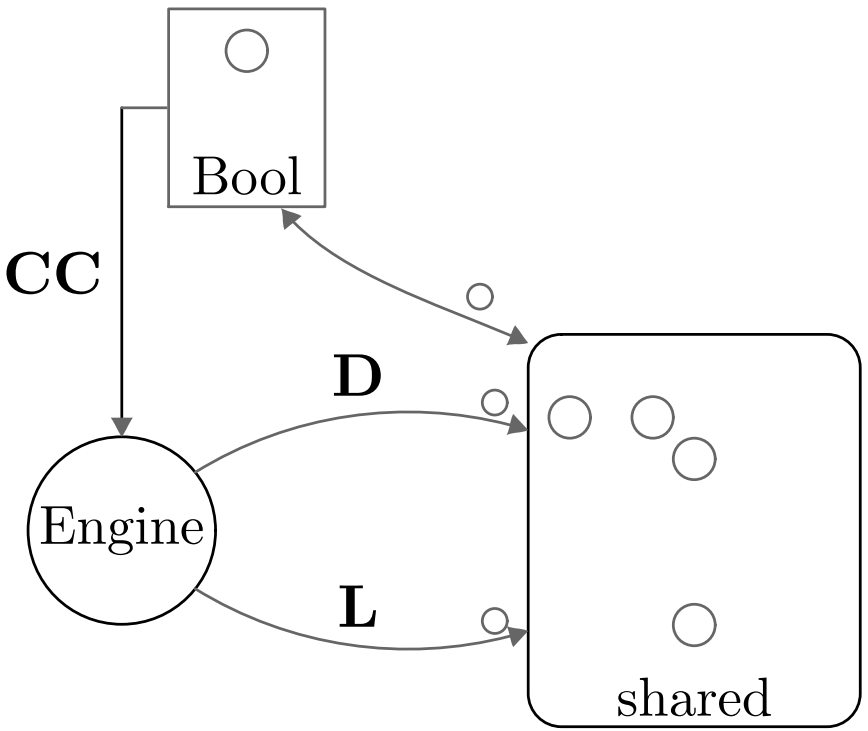
`Popop` one of the tools developped during the project.

SMT-solver: Big picture and CDCL



$$(A \vee B \implies C)$$
$$\wedge(C \implies D)$$
$$\wedge(D \wedge E \implies \neg A)$$

SMT-solver: Big picture and CDCL

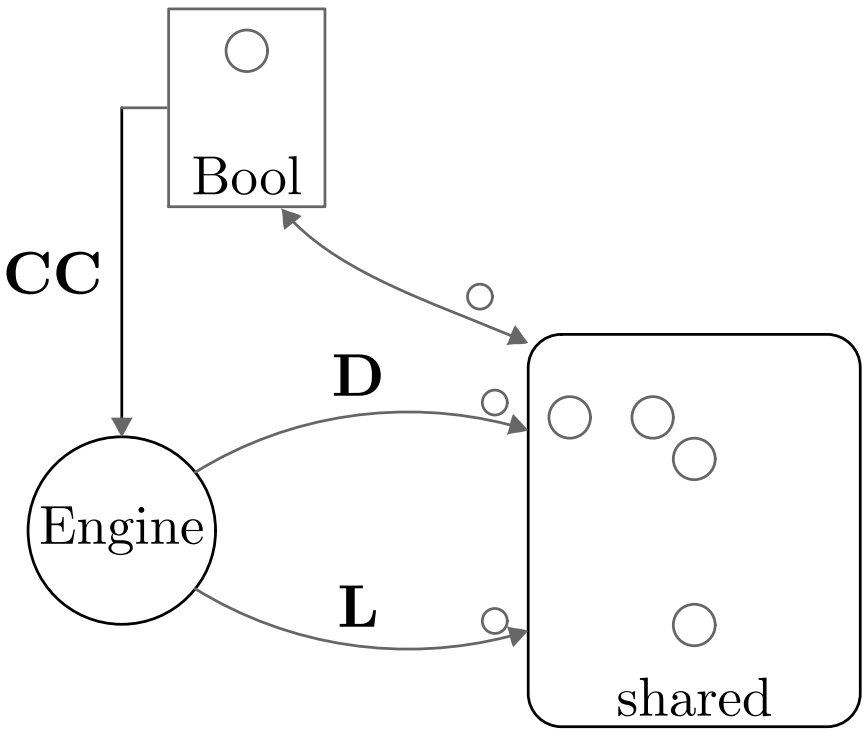


$$(A \vee B \implies C)$$
$$\wedge (C \implies D)$$
$$\wedge (D \wedge E \implies \neg A)$$

$$\frac{}{}$$

$$\frac{E}{}$$

SMT-solver: Big picture and CDCL

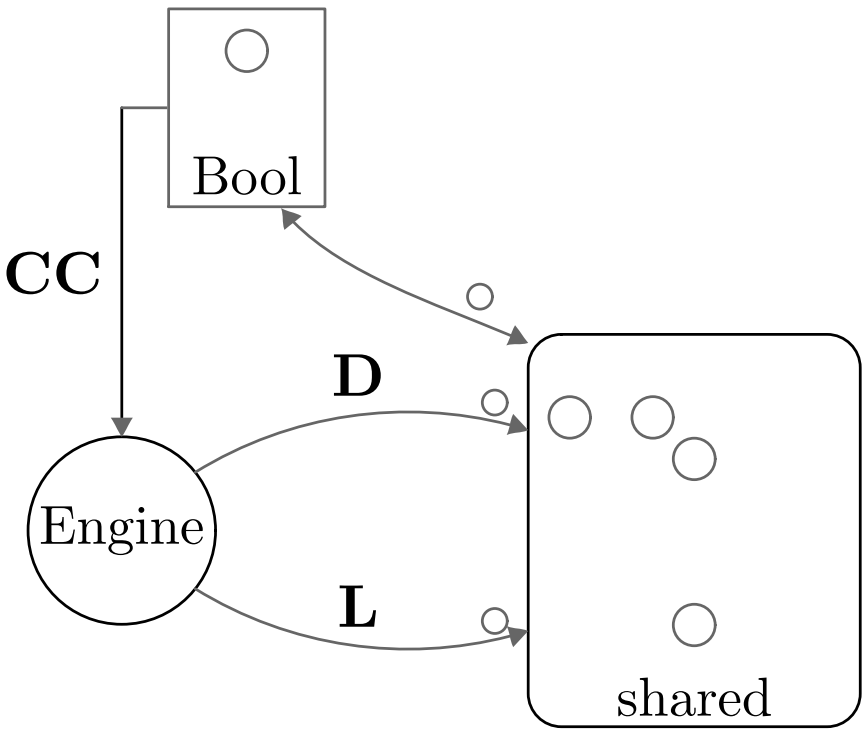


$$(A \vee B \implies C)$$
$$\wedge (C \implies D)$$
$$\wedge (D \wedge E \implies \neg A)$$

$$\frac{}{}$$

$$\frac{E}{}$$

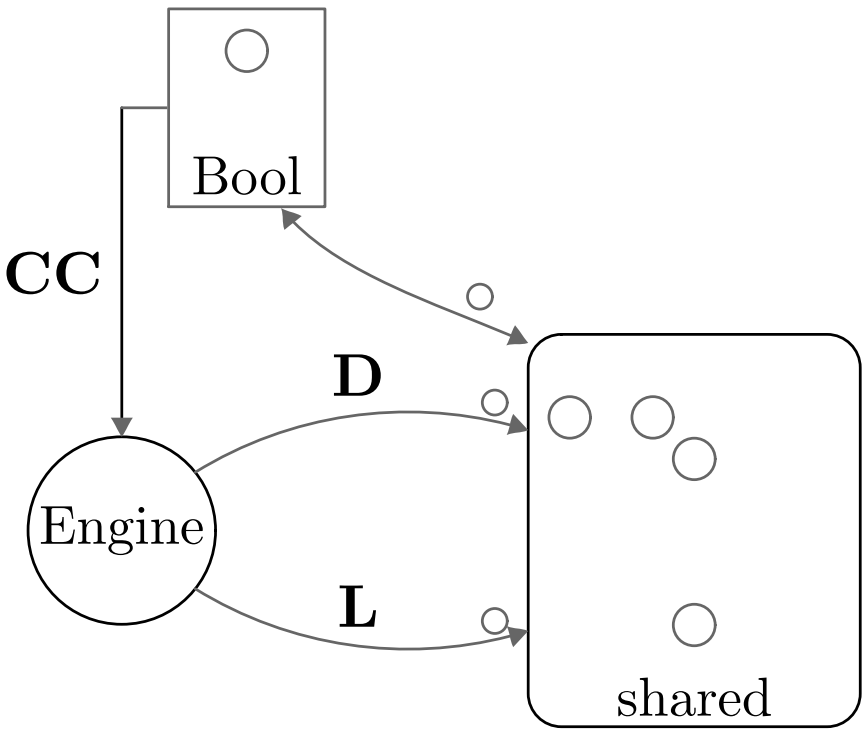
SMT-solver: Big picture and CDCL



$$(A \vee B \implies C)$$
$$\wedge (C \implies D)$$
$$\wedge (D \wedge E \implies \neg A)$$

$$\frac{\vdots}{E}$$

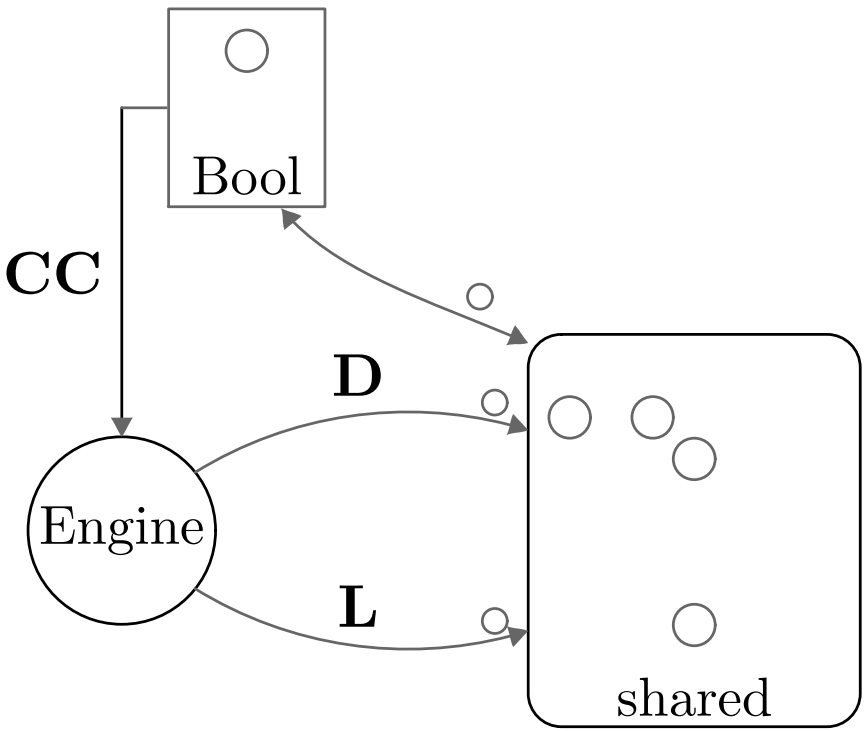
SMT-solver: Big picture and CDCL



$$(A \vee B \implies C)$$
$$\wedge (C \implies D)$$
$$\wedge (D \wedge E \implies \neg A)$$

$$\frac{\begin{array}{c} A_{\mathcal{D}} \\ \vdots \\ E \end{array}}{}$$

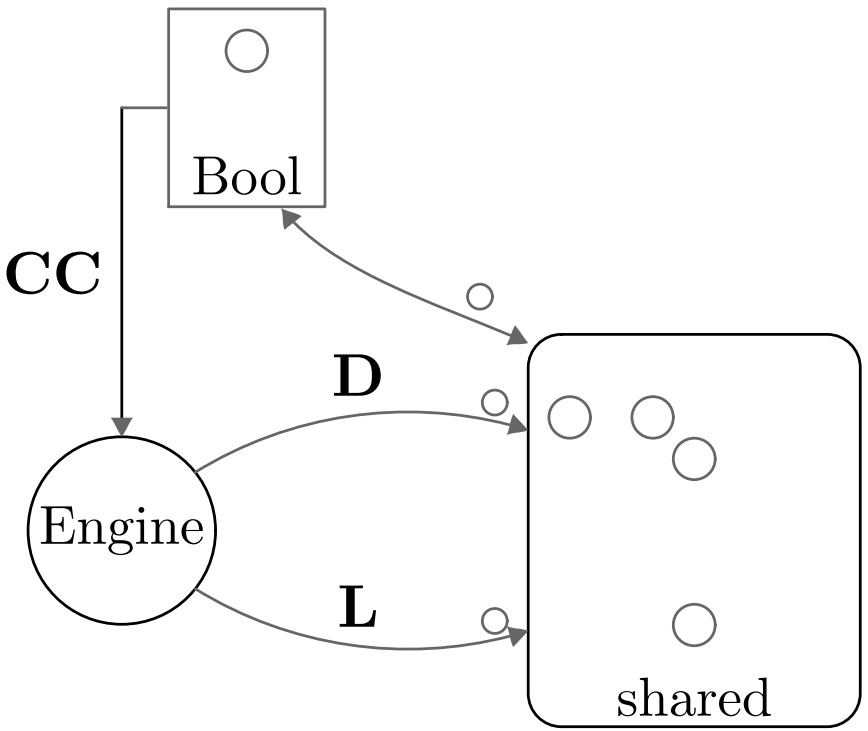
SMT-solver: Big picture and CDCL



$$(A \vee B \implies C)$$
$$\wedge (C \implies D)$$
$$\wedge (D \wedge E \implies \neg A)$$

$$\frac{\begin{array}{c} A_{\mathcal{D}} \\ \vdots \\ E \end{array}}{}$$

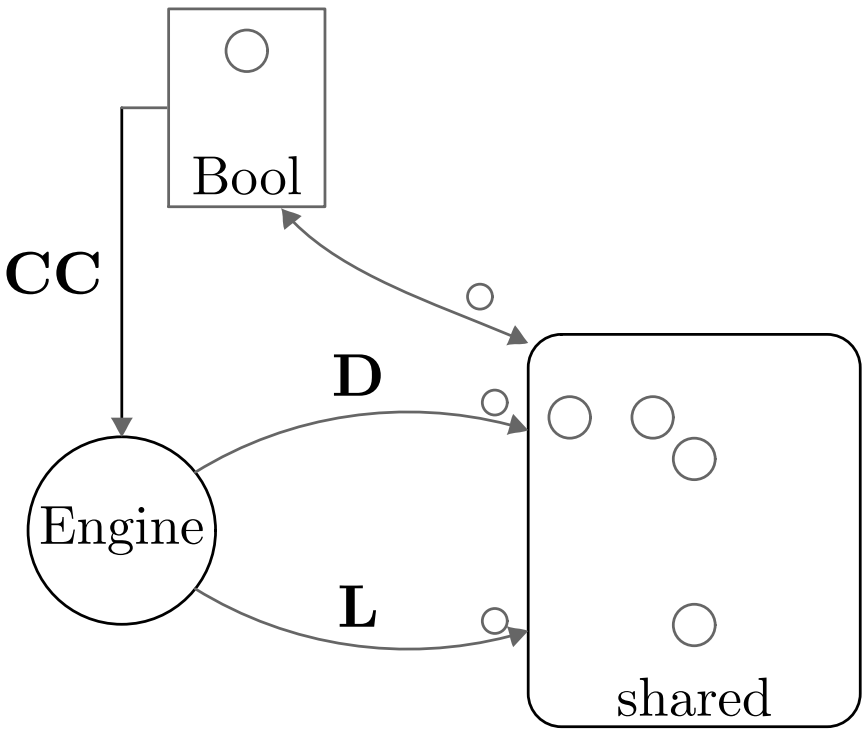
SMT-solver: Big picture and CDCL



$$(A \vee B \implies C)$$
$$\wedge (C \implies D)$$
$$\wedge (D \wedge E \implies \neg A)$$

$$\frac{\begin{array}{c} C \\ A_{\mathcal{D}} \\ \vdots \\ E \end{array}}{}$$

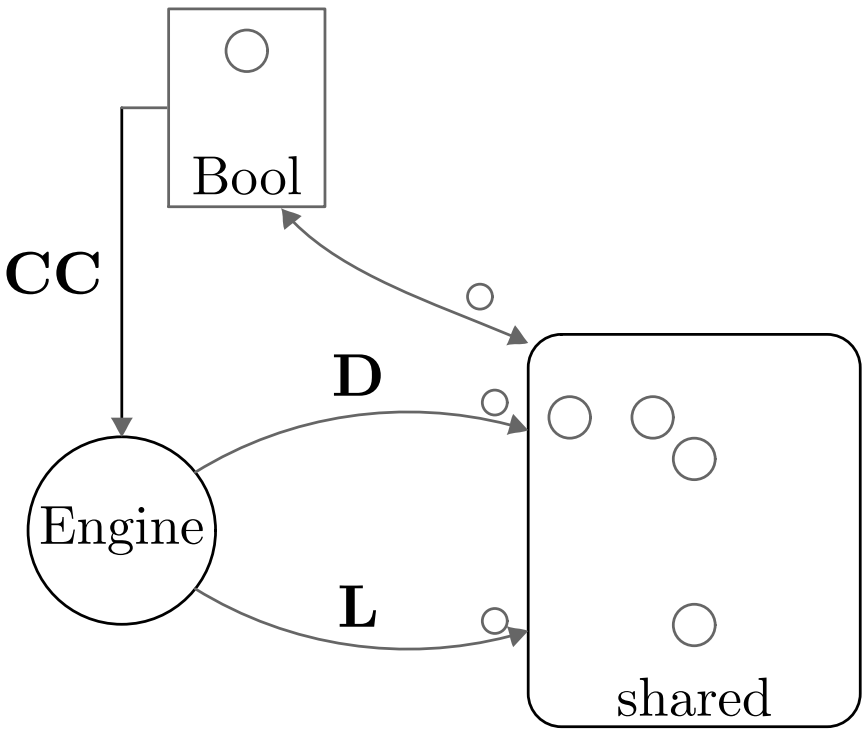
SMT-solver: Big picture and CDCL



$$(A \vee B \implies C)$$
$$\wedge (C \implies D)$$
$$\wedge (D \wedge E \implies \neg A)$$

$$\frac{\begin{array}{c} C \\ A_{\mathcal{D}} \\ \vdots \\ E \end{array}}{}$$

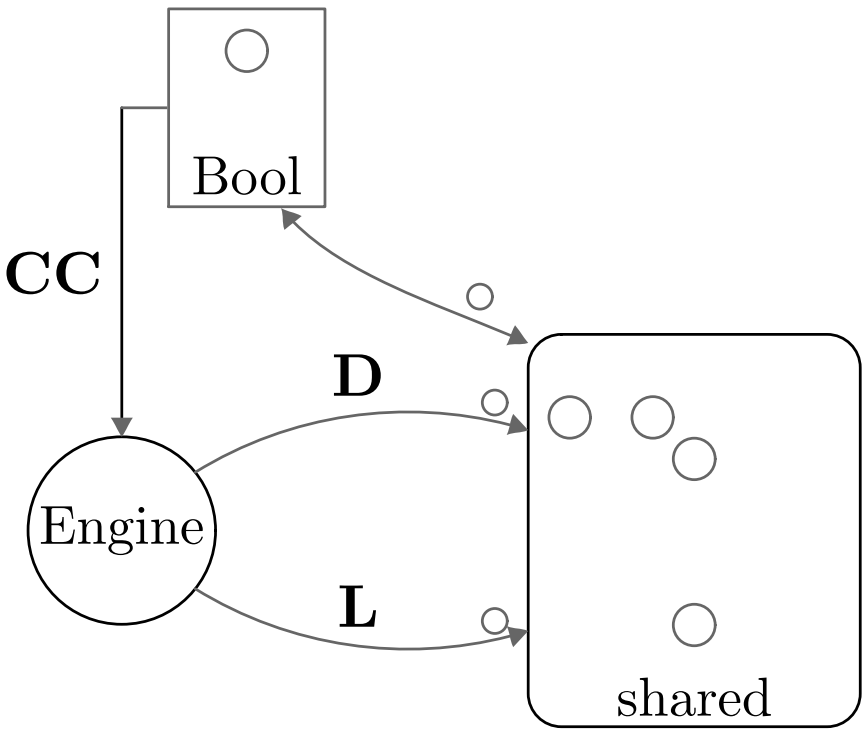
SMT-solver: Big picture and CDCL



$$(A \vee B \implies C)$$
$$\wedge (C \implies D)$$
$$\wedge (D \wedge E \implies \neg A)$$

$$\frac{}{\begin{array}{c} \neg D \\ C \\ A_{\mathcal{D}} \\ \vdots \\ E \end{array}}$$

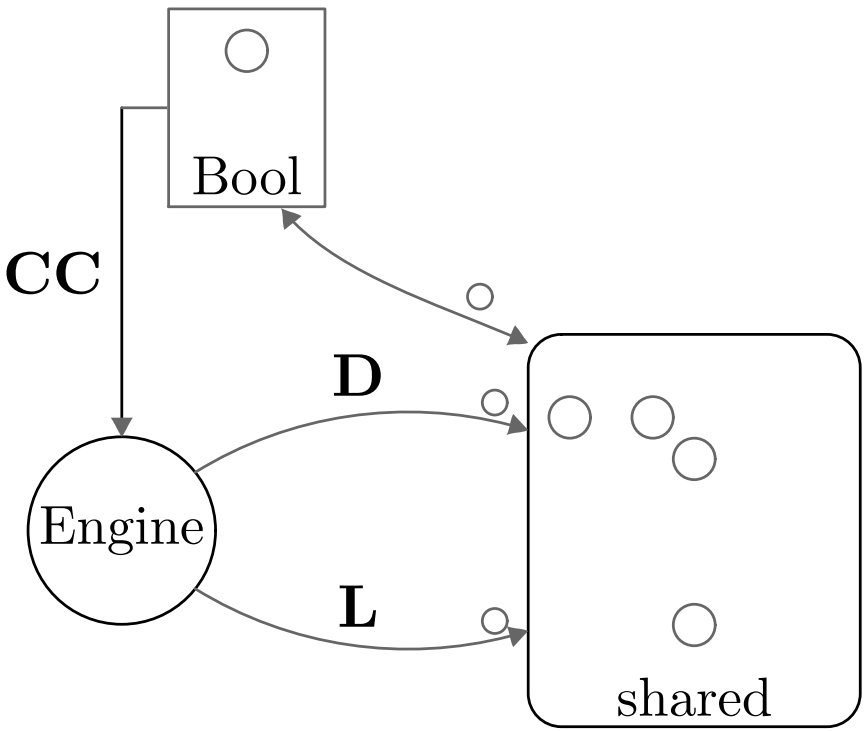
SMT-solver: Big picture and CDCL



$$(A \vee B \implies C)$$
$$\wedge(C \implies D)$$
$$\wedge(D \wedge E \implies \neg A)$$

$$\frac{}{\begin{array}{c}\neg D \\ C \\ A_{\mathcal{D}} \\ \vdots \\ E\end{array}}$$

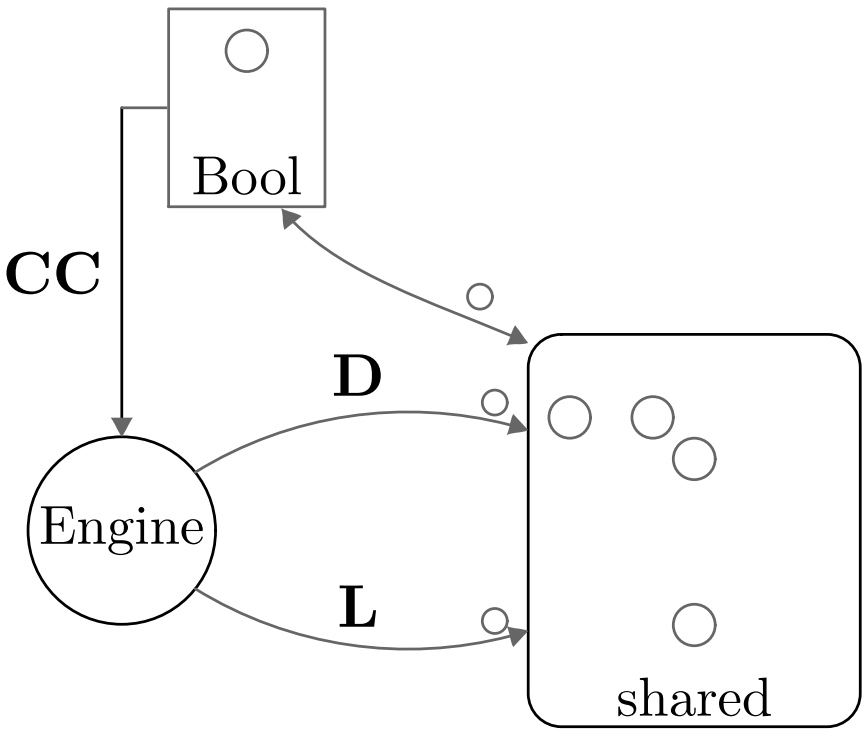
SMT-solver: Big picture and CDCL



$$(A \vee B \implies C)$$
$$\wedge(C \implies D)$$
$$\wedge(D \wedge E \implies \neg A)$$

$$\frac{}{\begin{array}{c}\neg D \\ C \\ A_{\mathcal{D}} \\ \vdots \\ E\end{array}}$$

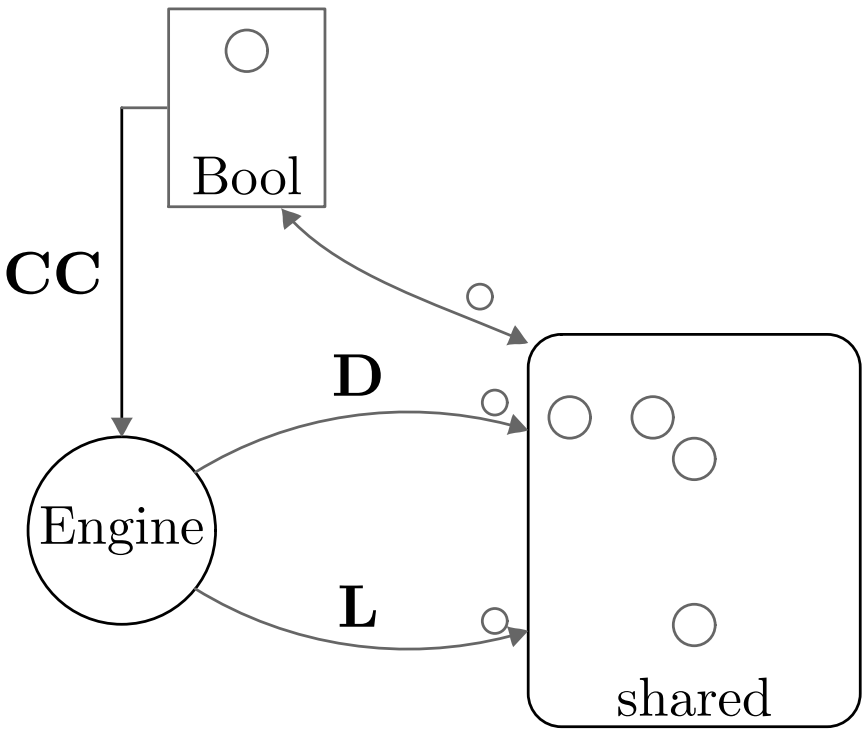
SMT-solver: Big picture and CDCL



$$(A \vee B \implies C)$$
$$\wedge(C \implies D)$$
$$\wedge(D \wedge E \implies \neg A)$$

$$\frac{\begin{array}{c}\bot \\ \neg D \\ C \\ A_{\mathcal{D}} \\ \vdots \\ E\end{array}}{}$$

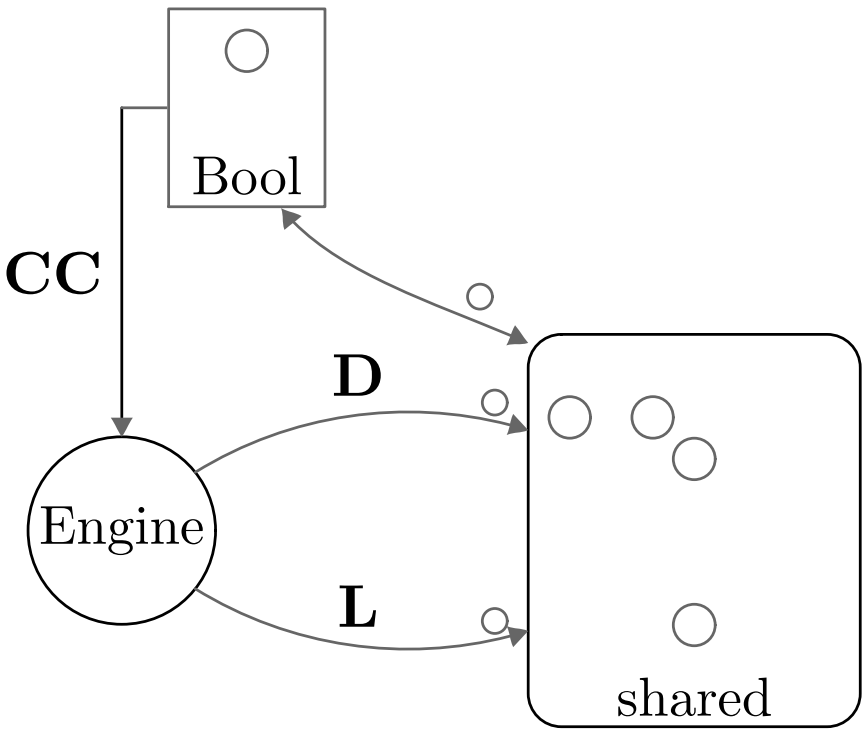
SMT-solver: Big picture and CDCL



$$(A \vee B \implies C)$$
$$\wedge (C \implies D)$$
$$\wedge (D \wedge E \implies \neg A)$$
$$\wedge (\neg E \vee \neg A)$$

$$\frac{}{E}$$

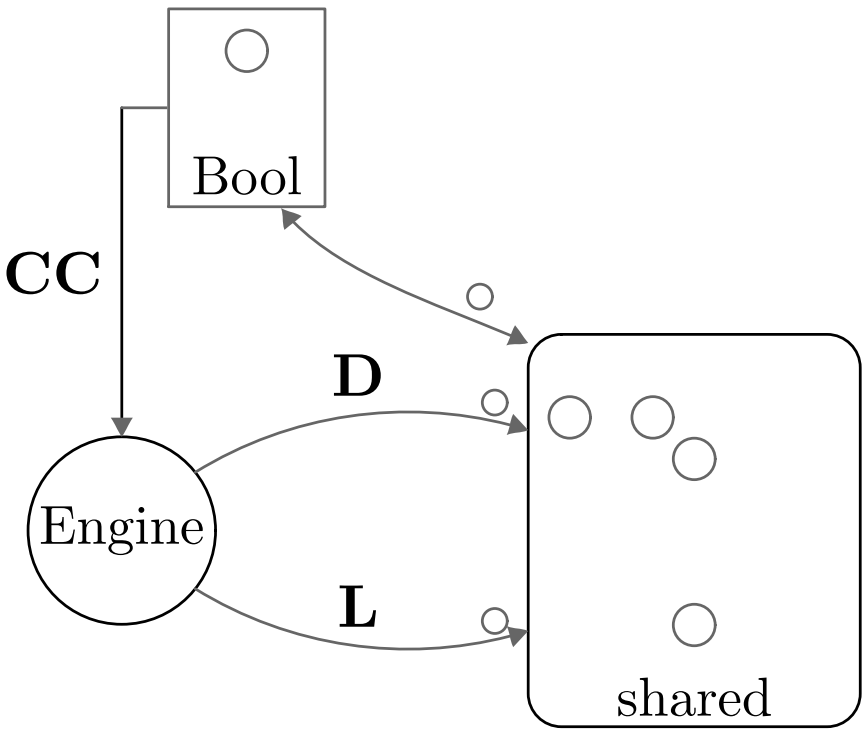
SMT-solver: Big picture and CDCL



$$(A \vee B \implies C)$$
$$\wedge (C \implies D)$$
$$\wedge (D \wedge E \implies \neg A)$$
$$\wedge (\neg E \vee \neg A)$$

$$E$$

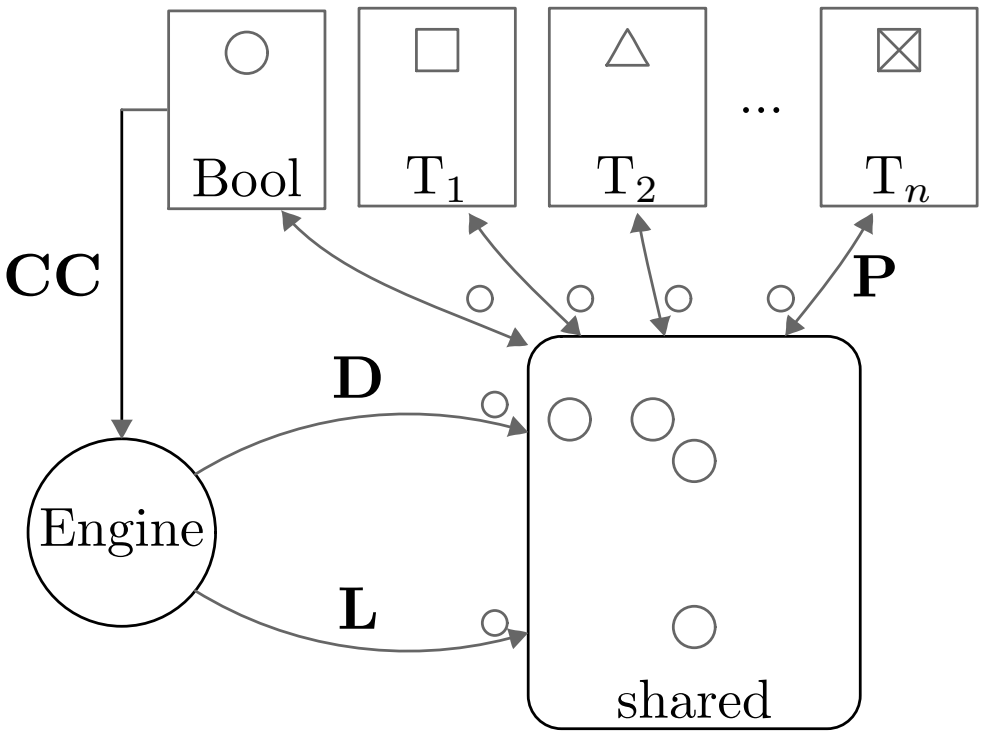
SMT-solver: Big picture and CDCL



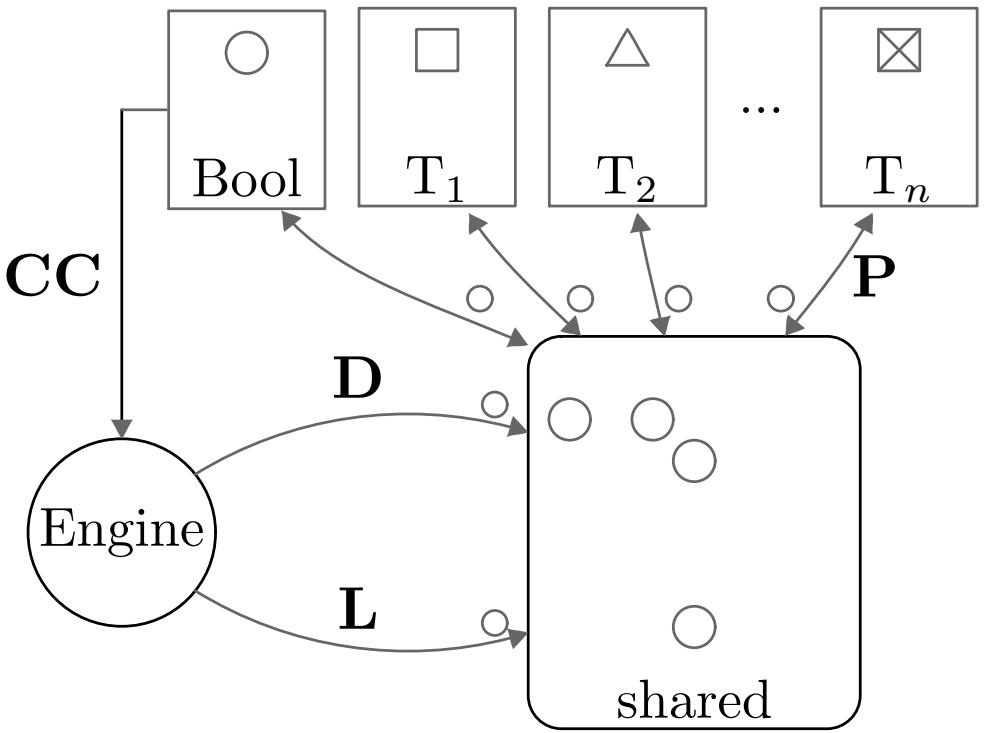
$$(A \vee B \implies C)$$
$$\wedge (C \implies D)$$
$$\wedge (D \wedge E \implies \neg A)$$
$$\wedge (\neg E \vee \neg A)$$

$$\neg A$$
$$E$$

SMT-solver: Big picture and CDCL

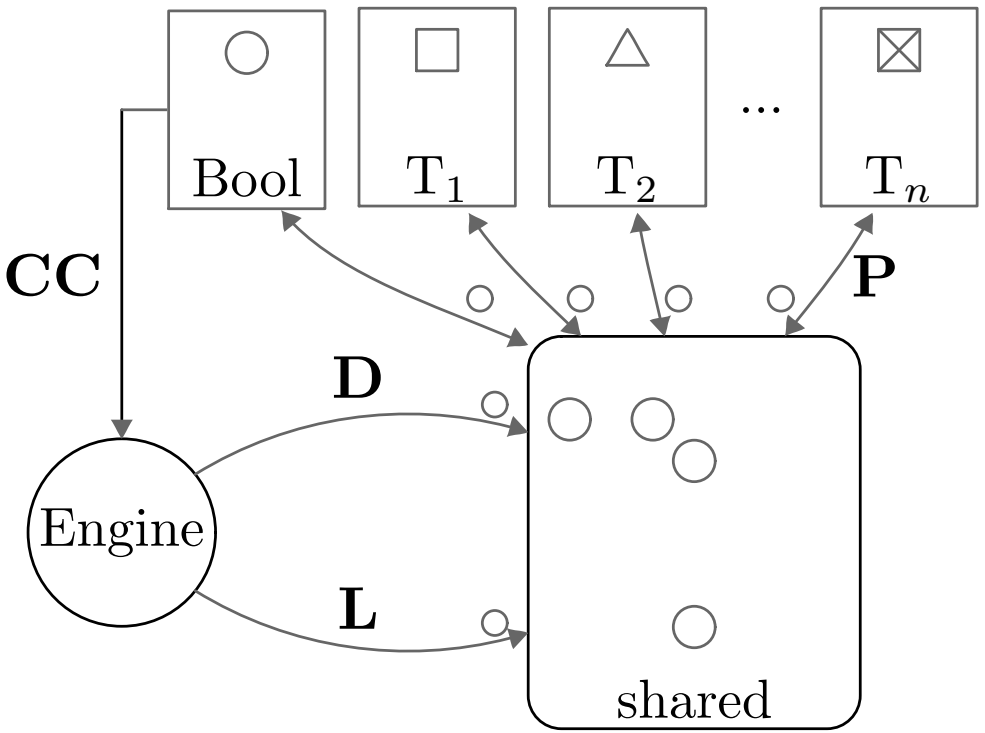


SMT-solver: Big picture and CDCL



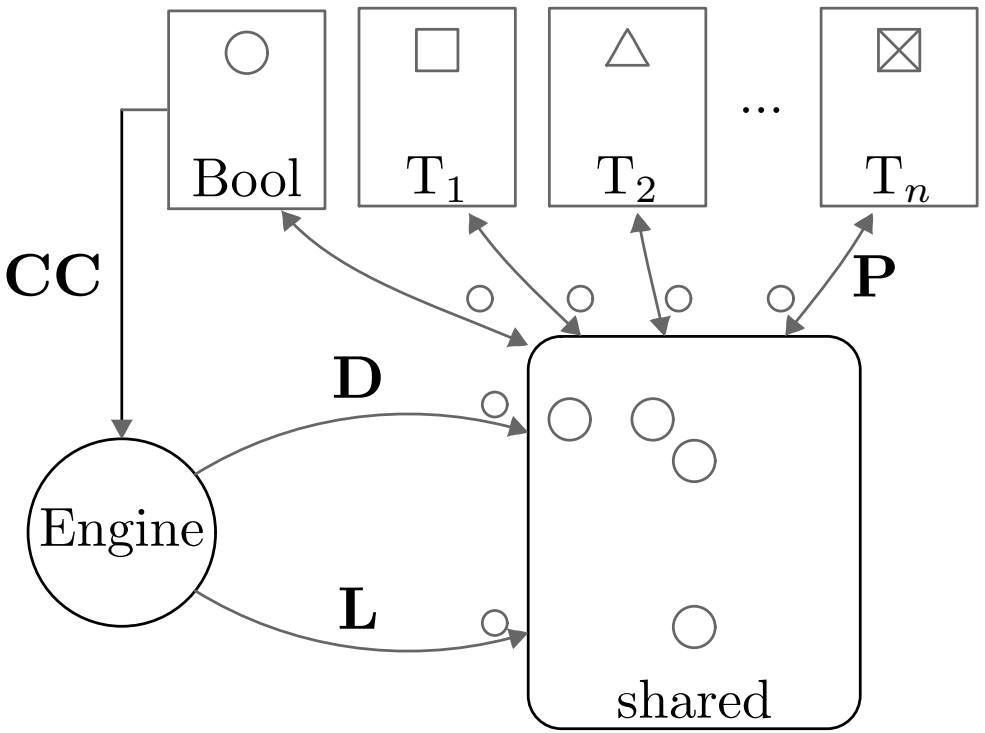
$$(A \vee B \implies C)$$
$$\wedge(C \implies D)$$
$$\wedge(D \wedge E \implies \neg A)$$

SMT-solver: Big picture and CDCL



$$\left(\begin{array}{c} 10 \leq x + t \vee 10 \leq y \\ \Longrightarrow\ 10 \leq z \end{array}\right)$$
$$\wedge\,(10 \leq z \implies t < 0)$$

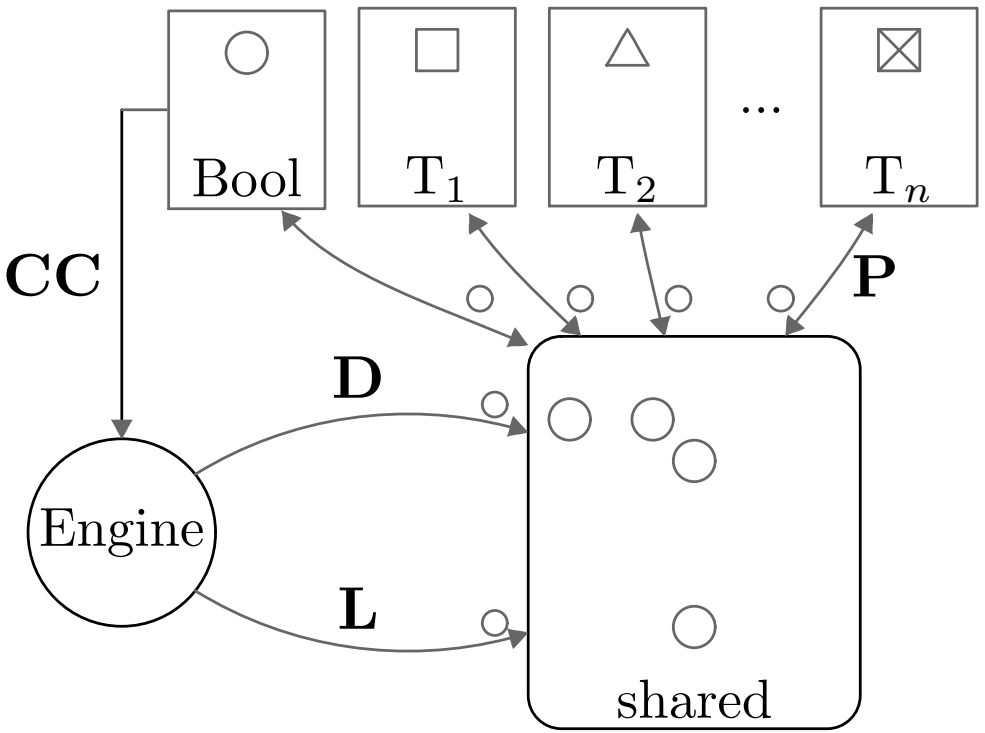
SMT-solver: Big picture and CDCL



$$\left(\begin{array}{c} 10 \leq x + t \vee 10 \leq y \\ \Longrightarrow\ 10 \leq z \end{array}\right)$$
$$\wedge\,(10 \leq z \implies t < 0)$$

$$(x \leq 0)$$

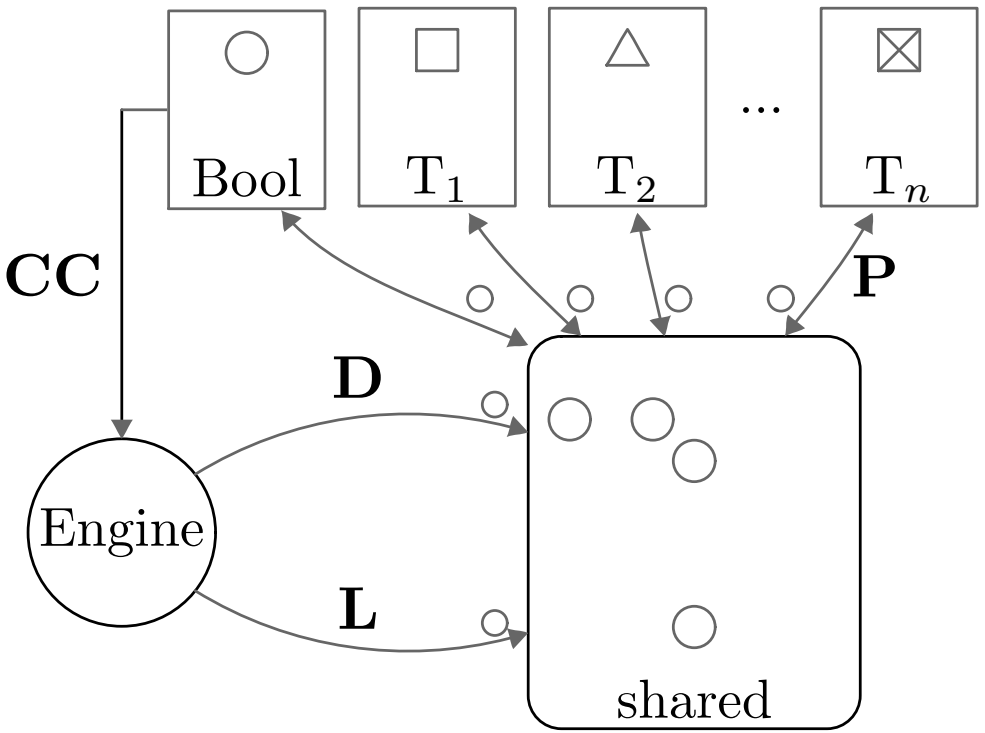
SMT-solver: Big picture and CDCL



$$\left(\begin{array}{c} 10 \leq x + t \vee 10 \leq y \\ \Longrightarrow\ 10 \leq z \end{array}\right)$$
$$\wedge\,(10 \leq z \implies t < 0)$$

$$\frac{\vdots}{(x \leq 0)}$$

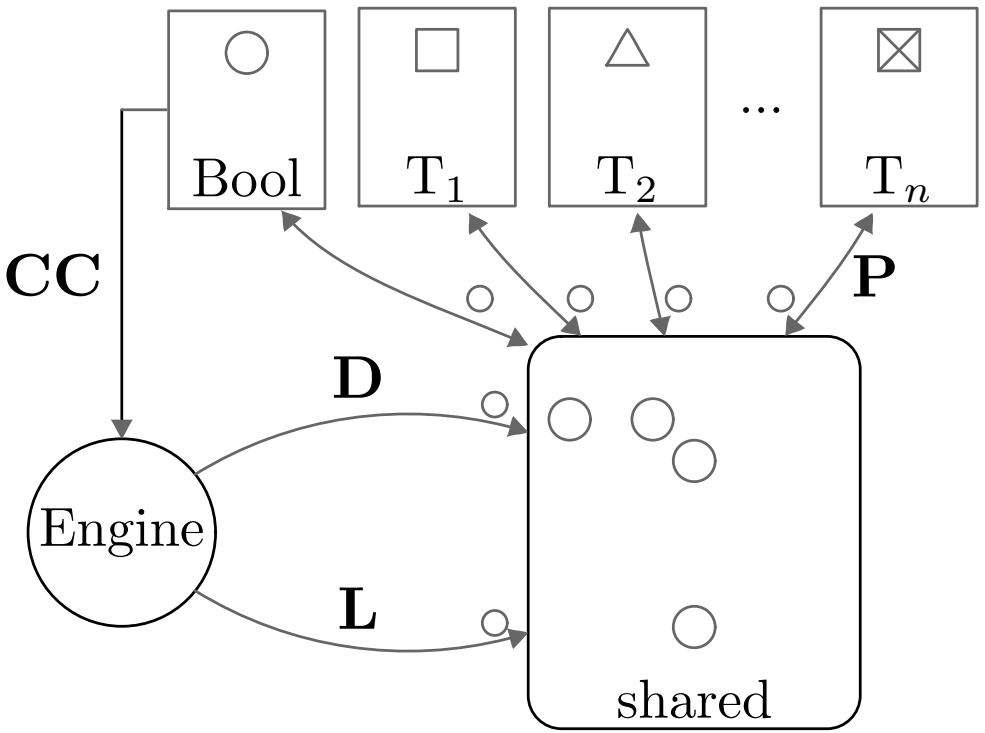
SMT-solver: Big picture and CDCL



$$\left(\begin{array}{c} 10 \leq x + t \vee 10 \leq y \\ \Longrightarrow\ 10 \leq z \end{array}\right)$$
$$\wedge\,(10 \leq z \implies t < 0)$$

$$\frac{\begin{array}{c}(10 \leq x + t)_{\mathcal{D}} \\ \vdots \\ (x \leq 0)\end{array}}{}$$

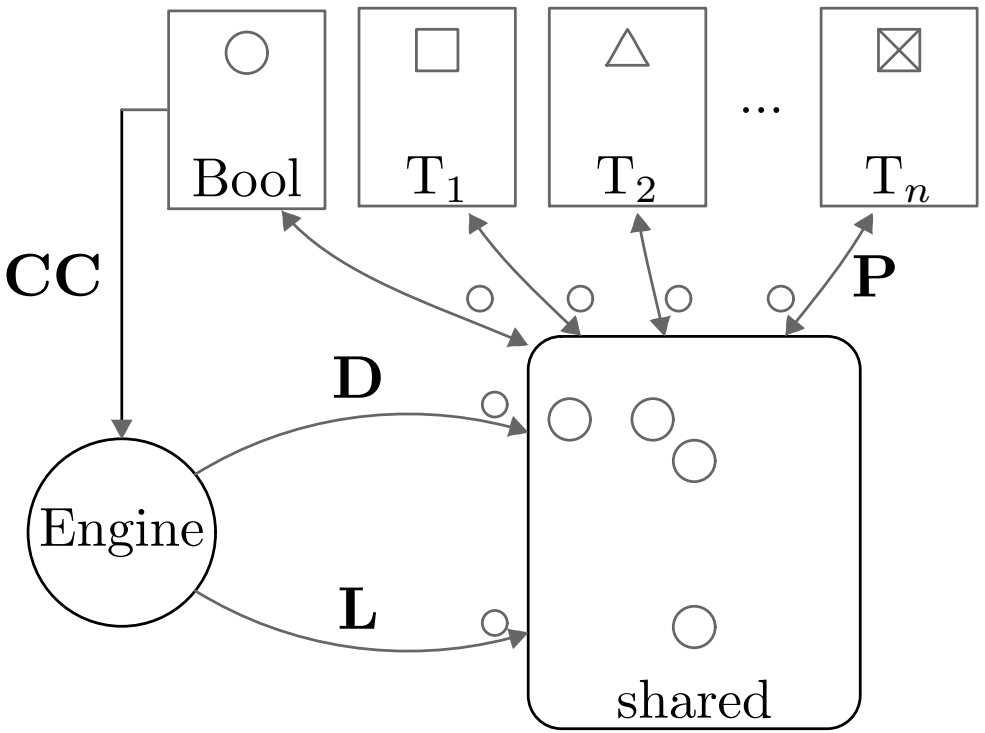
SMT-solver: Big picture and CDCL



$$\left(\begin{array}{c} 10 \leq x + t \vee 10 \leq y \\ \Longrightarrow\ 10 \leq z \end{array}\right) \wedge (10 \leq z \implies t < 0)$$

$$\frac{\begin{array}{c} 10 \leq z \\ (10 \leq x + t)_{\mathcal{D}} \\ \vdots \end{array}}{(x \leq 0)}$$

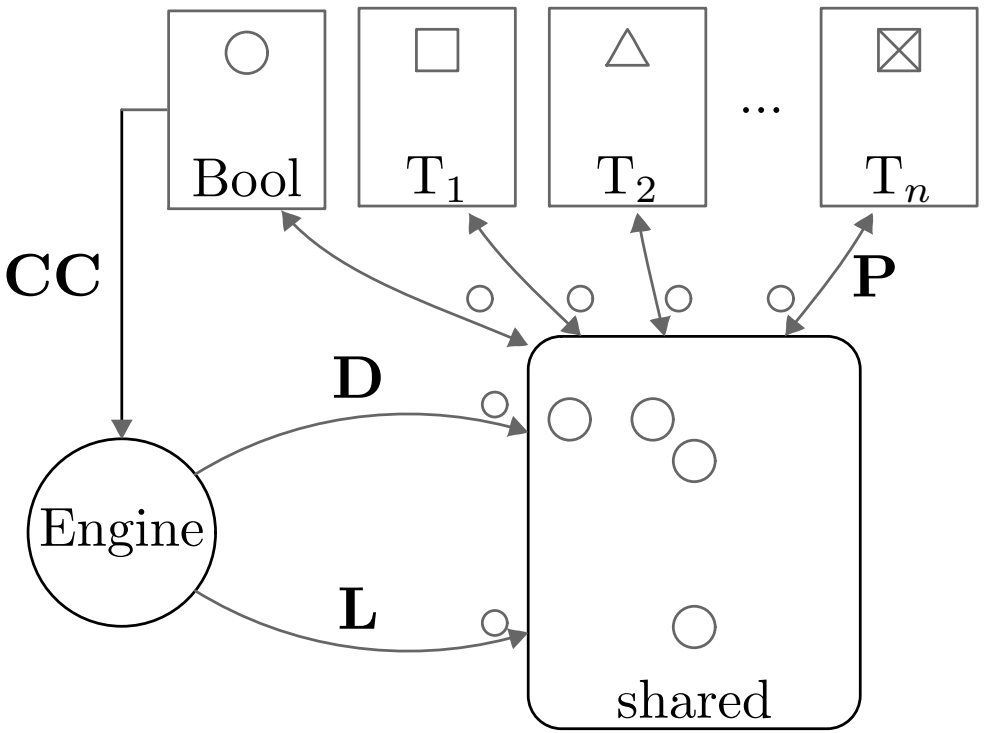
SMT-solver: Big picture and CDCL



$$\left(\begin{array}{c} 10 \leq x + t \vee 10 \leq y \\ \Longrightarrow\ 10 \leq z \end{array}\right)$$
$$\wedge\,(10 \leq z \implies t < 0)$$
$$\wedge\left(\begin{array}{c} t < 0 \wedge x \leq 0 \\ \Longrightarrow\ \neg 10 \leq x + t \end{array}\right)$$

$$\frac{\begin{array}{c} \neg(t < 0) \\ 10 \leq z \\ (10 \leq x + t)_{\mathcal{D}} \\ \vdots \\ (x \leq 0) \end{array}}{}$$

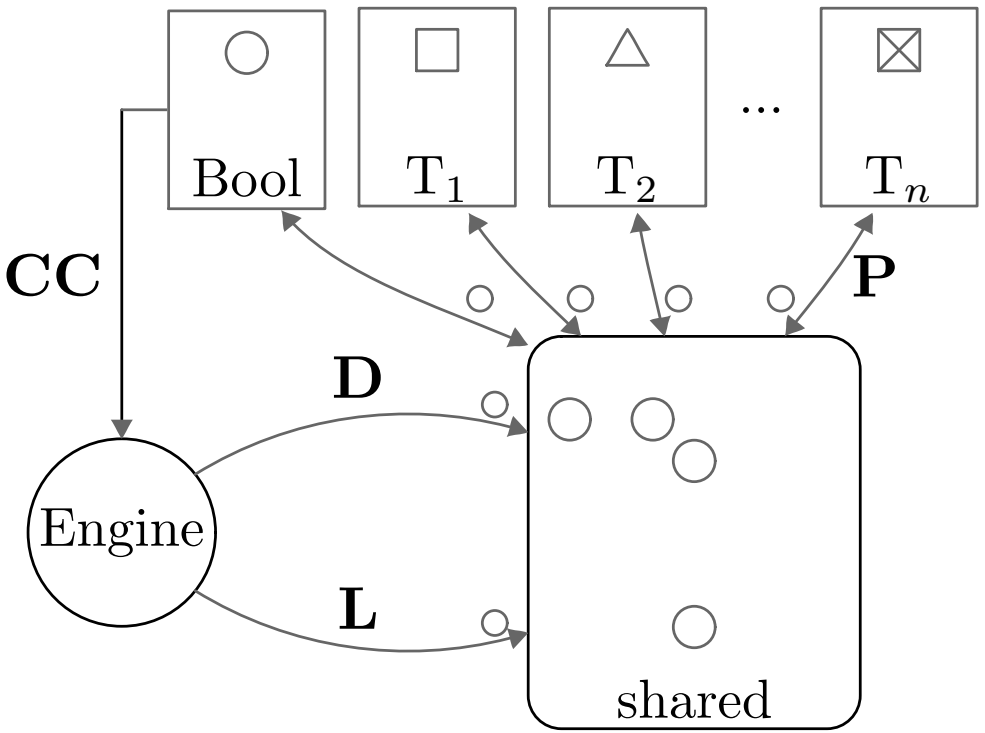
SMT-solver: Big picture and CDCL



$$\left(\begin{array}{c} 10 \leq x + t \vee 10 \leq y \\ \Longrightarrow\ 10 \leq z \end{array}\right)$$
$$\wedge\,(10 \leq z \implies t < 0)$$
$$\wedge\left(\begin{array}{c} t < 0 \wedge x \leq 0 \\ \Longrightarrow\ \neg 10 \leq x + t \end{array}\right)$$

$$\frac{\begin{array}{c} \neg(t < 0) \\ 10 \leq z \\ (10 \leq x + t)_{\mathcal{D}} \\ \vdots \\ (x \leq 0) \end{array}}{}$$

SMT-solver: Big picture and CDCL



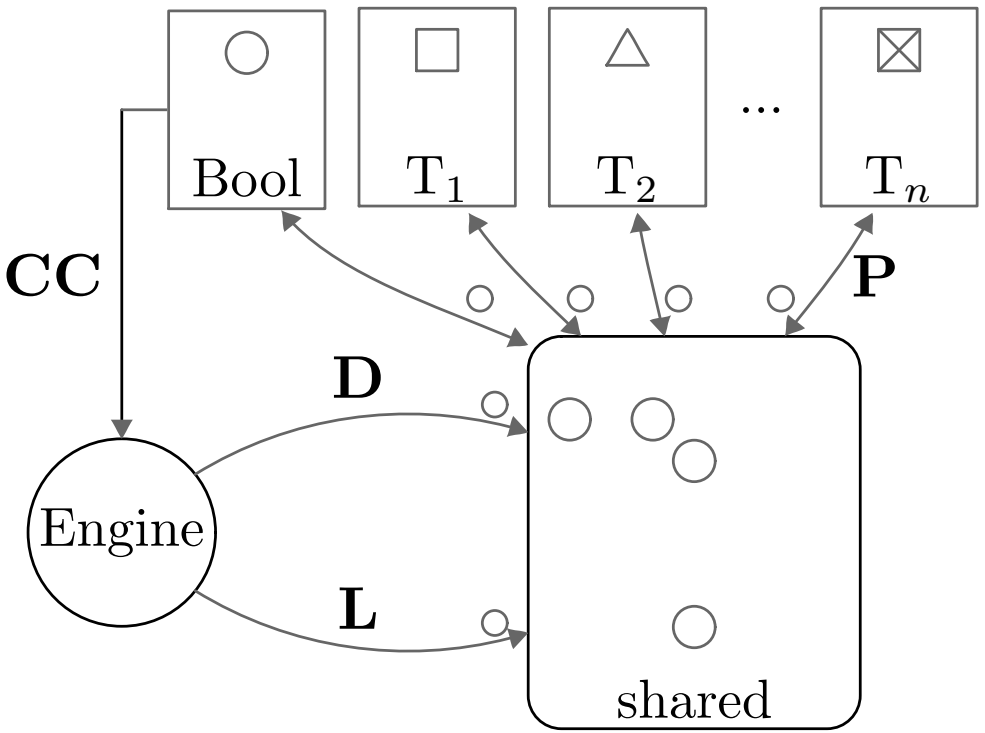
$$\left(\begin{array}{c} 10 \leq x + t \vee 10 \leq y \\ \Longrightarrow \; 10 \leq z \end{array}\right)$$

$$\wedge\,(10 \leq z \implies t < 0)$$

$$\wedge(\neg x \leq \; 0 \vee \neg 10 \leq x + t)$$

$$(x \leq 0)$$

SMT-solver: Big picture and CDCL



$$\left(\begin{array}{c} 10 \leq x + t \vee 10 \leq y \\ \Longrightarrow\ 10 \leq z \end{array}\right)$$

$$\wedge\,(10 \leq z \implies t < 0)$$

$$\wedge(\neg x \leq 10 \vee \neg 10 \leq x + t)$$

$$(x \leq 0)$$

SAT-solver: CDCL

- Decision: an unset variable is added to the partial model
- Propagation: propagate values using the constraints
- Conflict: a constraint is unsatisfiable in the partial model
- Learning: compute a constraint which:
  1. is implied by the constraints
  2. is unsatisfiable in the partial model
  3. disallows the last decision by propagation

SAT-solver: CDCL

- Decision: an unset variable is added to the partial model
- Propagation: propagate values using the constraints
- Conflict: a constraint is unsatisfiable in the partial model
- Learning: compute a constraint which:
  1. is implied by the constraints
  2. is unsatisfiable in the partial model
  3. disallows the last decision by propagation

SAT-solver: CDCL

- Decision: an unset variable is added to the partial model
- Propagation: propagate values using the constraints
- Conflict: a constraint is unsatisfiable in the partial model
- Learning: compute a constraint which:
  1. is implied by the constraints
  2. is unsatisfiable in the partial model
  3. disallows the last decision by propagation

SAT-solver: CDCL

- Decision: an unset variable is added to the partial model
- Propagation: propagate values using the constraints
- Conflict: a constraint is unsatisfiable in the partial model
- Learning: compute a constraint which:
  1. is implied by the constraints
  2. is unsatisfiable in the partial model
  3. disallows the last decision by propagation

SAT-solver: CDCL

- Decision: an unset variable is added to the partial model
- Propagation: propagate values using the constraints
- Conflict: a constraint is unsatisfiable in the partial model
- Learning: compute a constraint which:
  1. is implied by the constraints
  2. is unsatisfiable in the partial model
  3. disallows the last decision by propagation

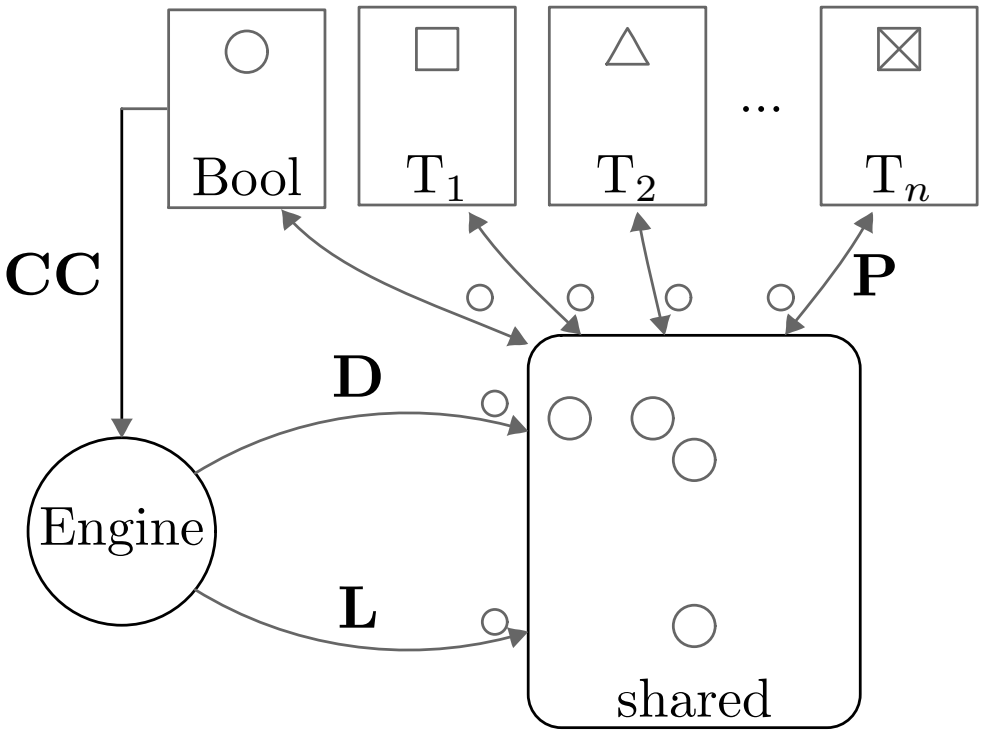
SAT-solver: CDCL

- Decision: an unset variable is added to the partial model
- Propagation: propagate values using the constraints
- Conflict: a constraint is unsatisfiable in the partial model
- Learning: compute a constraint which:
  1. is implied by the constraints
  2. is unsatisfiable in the partial model
  3. disallows the last decision by propagation

SAT-solver: CDCL

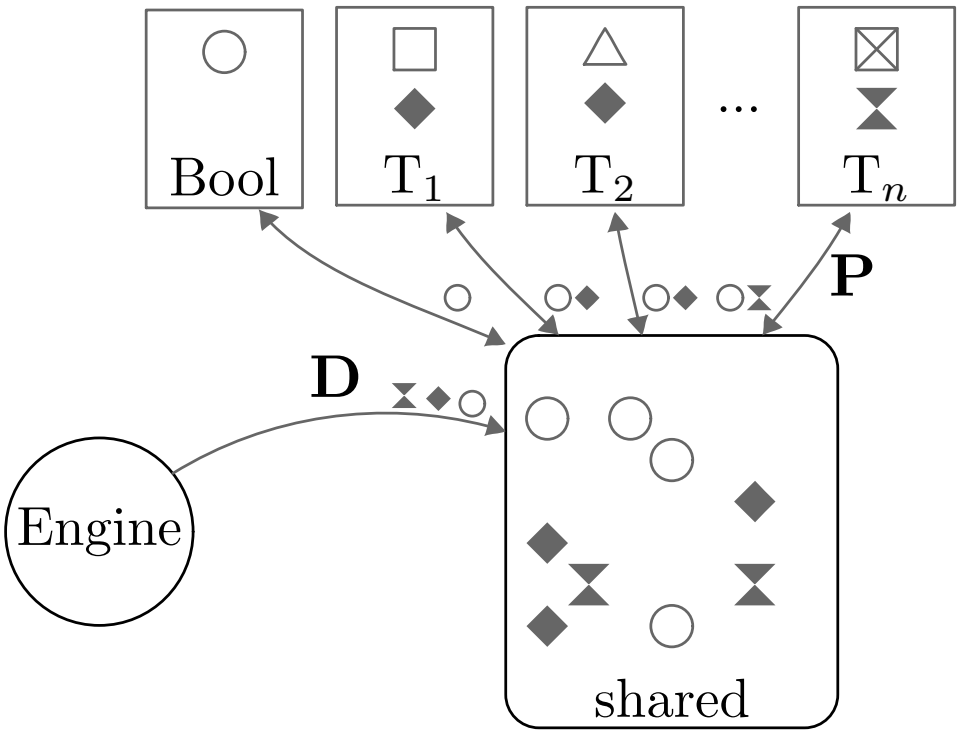
- Decision: an unset variable is added to the partial model
- Propagation: propagate values using the constraints
- Conflict: a constraint is unsatisfiable in the partial model
- Learning: compute a constraint which:
  1. is implied by the constraints
  2. is unsatisfiable in the partial model
  3. disallows the last decision by propagation

SMT-solver: Difficulties



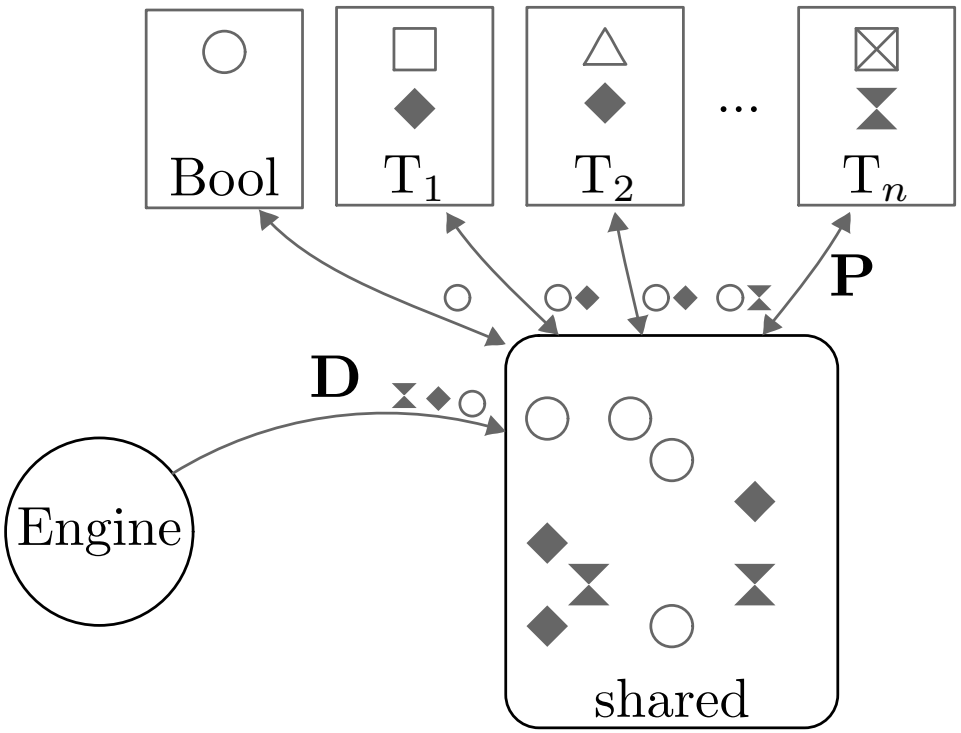
- communication using only booleans
- split-on-demand: $x \leq 11 \vee 11 < x$
- learning: no new literals

Big picture



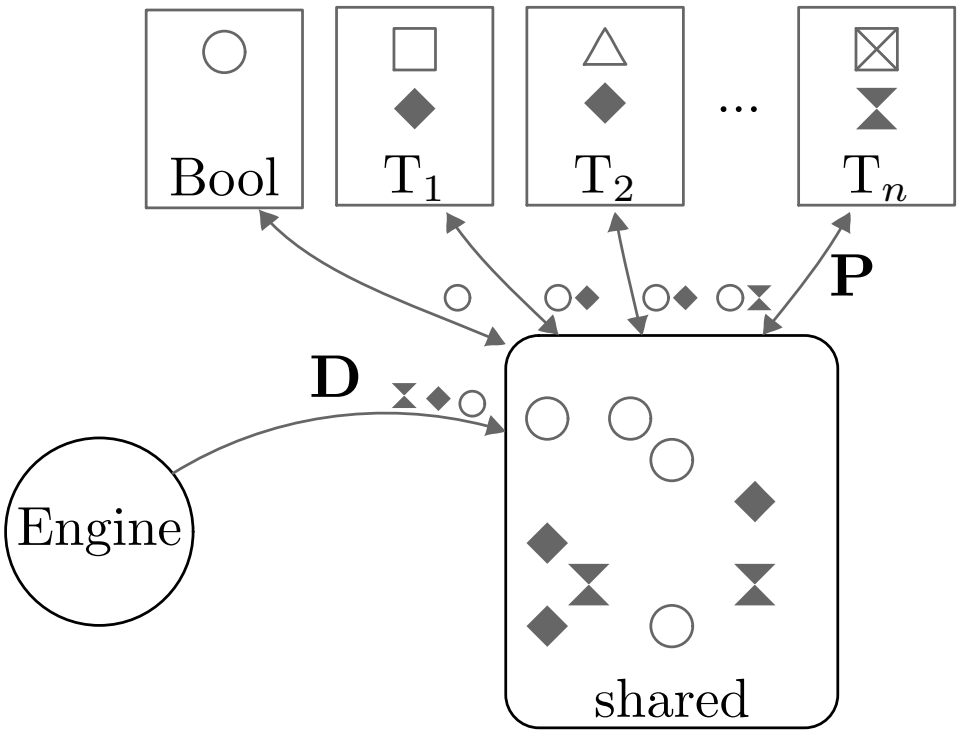
- Specific domains
- Precise and numerous propagator
- Easy model computation

Big picture



- Specific domains
- Precise and numerous propagator
- Easy model computation

Big picture



- Specific domains
- Precise and numerous propagator
- Easy model computation

Propagation and domains

- arithmetic intervals: $\mathcal{D}_I(x) = [\![1; 3]\!]$ and $\mathcal{D}_I(x + y) = [\![2; 4]\!]$ implies $\mathcal{D}_I(y) = [\![-1; 3]\!]$
- intervals and modulos: reduce product (AI)
- intervals and arrays: $\text{select}(x, \text{store}(\text{store}(\text{store}(\text{const}(4), y_3, 3), y_2, 2), y_1, 1)) = t$ $\mathcal{D}_I(t) = [\![1; 4]\!]$
- bitvectors and arithmetic intervals: $\mathcal{D}_I(x) = [\![0; +\infty[$ implies $\mathcal{D}_I(x \mid 31) = [\![2^{32}; +\infty]\!]$
- floating-points

Propagation and domains

- arithmetic intervals: $\mathcal{D}_I(x) = [\![1; 3]\!]$ and $\mathcal{D}_I(x + y) = [\![2; 4]\!]$ implies $\mathcal{D}_I(y) = [\![-1; 3]\!]$
- intervals and modulos: reduce product (AI)
- intervals and arrays: $\text{select}(x, \text{store}(\text{store}(\text{store}(\text{const}(4), y_3, 3), y_2, 2), y_1, 1)) = t$ $\mathcal{D}_I(t) = [\![1; 4]\!]$
- bitvectors and arithmetic intervals: $\mathcal{D}_I(x) = [\![0; +\infty[$ implies $\mathcal{D}_I(x \mid 31) = [\![2^{32}; +\infty]\!]$
- floating-points

Propagation and domains

- arithmetic intervals: $\mathcal{D}_I(x) = [\![1; 3]\!]$ and $\mathcal{D}_I(x + y) = [\![2; 4]\!]$ implies $\mathcal{D}_I(y) = [\![-1; 3]\!]$
- intervals and modulos: reduce product (AI)
- intervals and arrays:
  $\text{select}(x, \text{store}(\text{store}(\text{store}(\text{const}(4), y_3, 3), y_2, 2), y_1, 1)) = t$
  $\mathcal{D}_I(t) = [\![1; 4]\!]$
- bitvectors and arithmetic intervals: $\mathcal{D}_I(x) = [\![0; +\infty[$ implies $\mathcal{D}_I(x \mid 31) = [\![2^{32}; +\infty]\!]$
- floating-points

Propagation and domains

- arithmetic intervals: $\mathcal{D}_I(x) = [\![1; 3]\!]$ and $\mathcal{D}_I(x + y) = [\![2; 4]\!]$ implies $\mathcal{D}_I(y) = [\![-1; 3]\!]$
- intervals and modulos: reduce product (AI)
- intervals and arrays:
  $\text{select}(x, \text{store}(\text{store}(\text{store}(\text{const}(4), y_3, 3), y_2, 2), y_1, 1)) = t$
  $\mathcal{D}_I(t) = [\![1; 4]\!]$
- bitvectors and arithmetic intervals: $\mathcal{D}_I(x) = [\![0; +\infty[$ implies $\mathcal{D}_I(x \mid 31) = [\![2^{32}; +\infty]\!]$
- floating-points

Propagation and domains

- arithmetic intervals: $\mathcal{D}_I(x) = [\![1; 3]\!]$ and $\mathcal{D}_I(x + y) = [\![2; 4]\!]$ implies $\mathcal{D}_I(y) = [\![-1; 3]\!]$
- intervals and modulos: reduce product (AI)
- intervals and arrays:
  $\text{select}(x, \text{store}(\text{store}(\text{store}(\text{const}(4), y_3, 3), y_2, 2), y_1, 1)) = t$
  $\mathcal{D}_I(t) = [\![1; 4]\!]$
- bitvectors and arithmetic intervals: $\mathcal{D}_I(x) = [\![0; +\infty[$ implies $\mathcal{D}_I(x \mid 31) = [\![2^{32}; +\infty]\!]$
- floating-points

Global constraints

- DBM: Difference Constraint
- Simplex when slow convergence appears
- ...

Global constraints

- DBM: Difference Constraint
- Simplex when slow convergence appears
- ...

Global constraints

- DBM: Difference Constraint
- Simplex when slow convergence appears
- ...

Difficulties

- No backjumping
- No learning
- Costly propagation

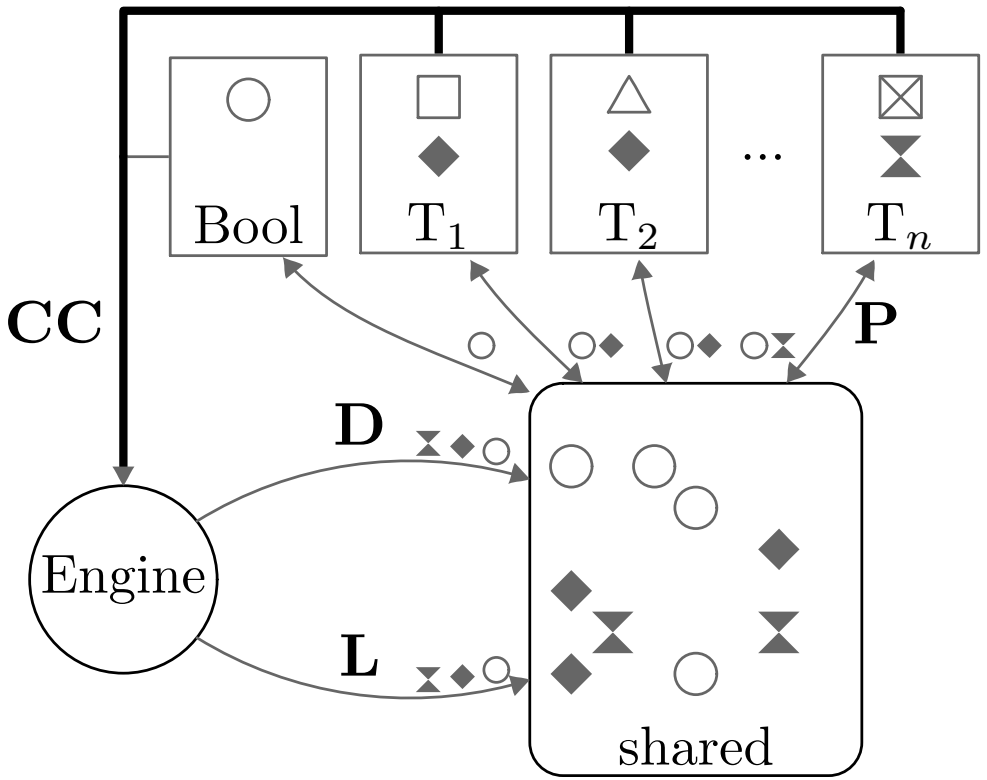
Difficulties

- No backjumping
- No learning
- Costly propagation

Difficulties

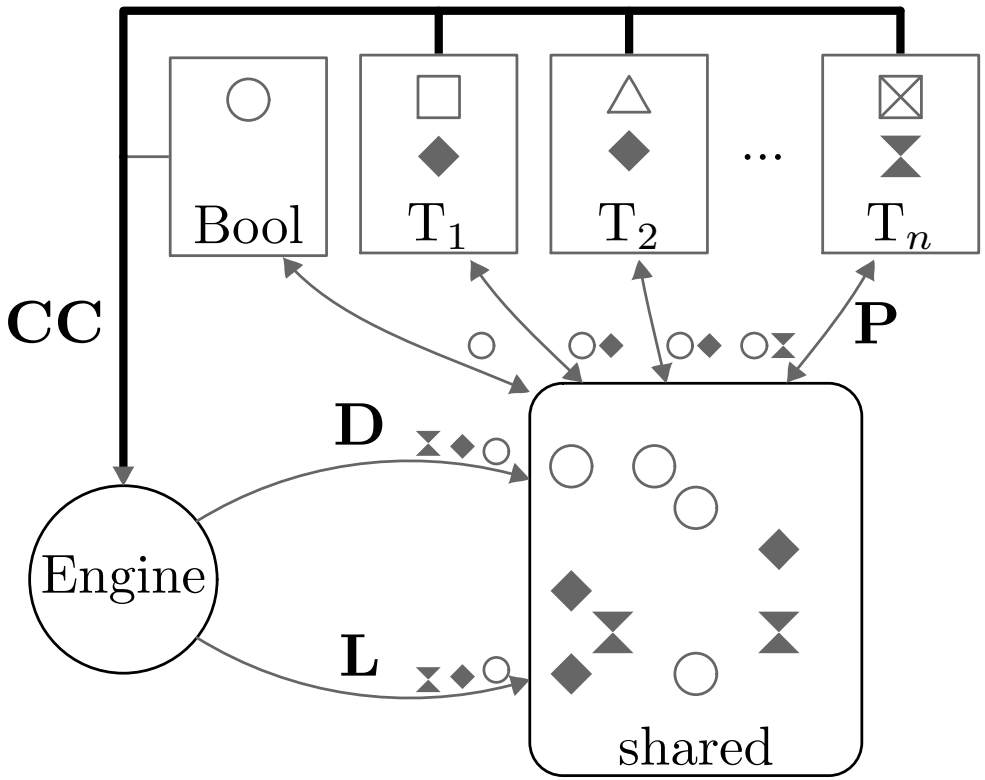
- No backjumping
- No learning
- Costly propagation

Ideas[Generalisation of de Moura (2013)]



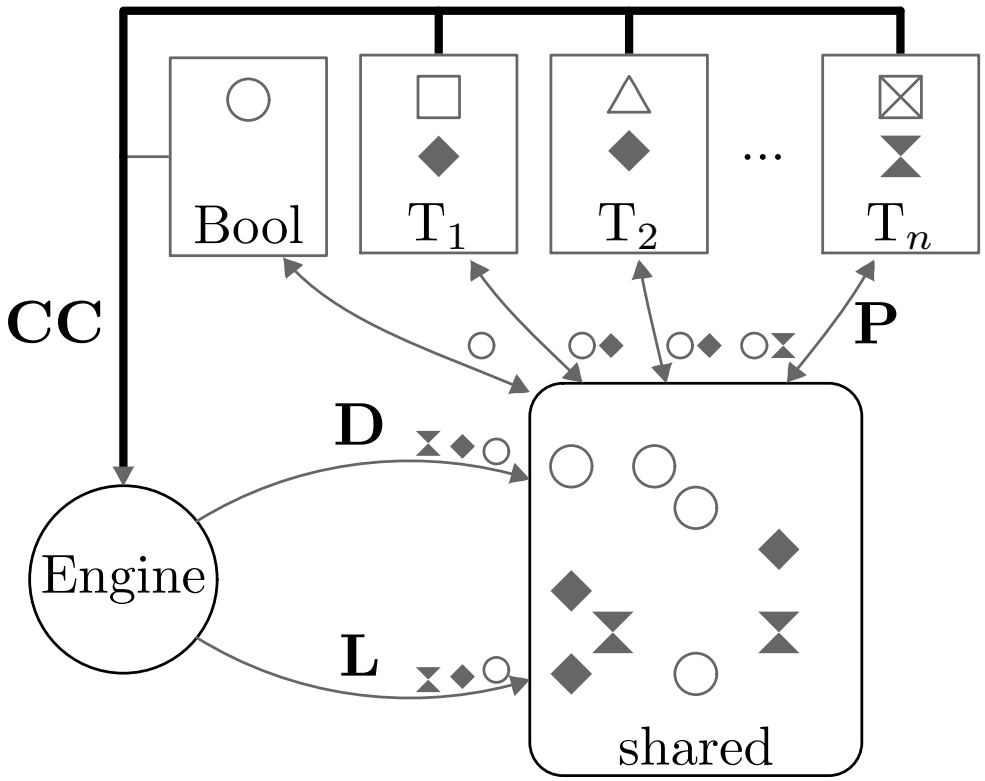
- Can decide on every variables
- Propagate using specific domains
- Learn new constraints

Ideas[Generalisation of de Moura (2013)]



- Can decide on every variables
- Propagate using specific domains
- Learn new constraints

Ideas[Generalisation of de Moura (2013)]



- Can decide on every variables
- Propagate using specific domains
- Learn new constraints

Example: Arithmetic

$$10 \leq x - y \quad \wedge \quad 10 < x + y + z$$

1. Decide $x = 0$
2. Propagate using $x - y$
3. Conflict in $x + y + z$
4. Compute clause conflict
5. Propagate

	$\mathcal{D}$	*reason*
x	$]-\infty;+\infty[$	
y	$]-\infty;+\infty[$	
z	$]-\infty;0]$	...
x-y	$[10;+\infty[$	
x+y+z	$]10;+\infty[$	

Example: Arithmetic

$$10 \leq x - y \quad \wedge \quad 10 < x + y + z$$

1. Decide $x = 0$
2. Propagate using $x - y$
3. Conflict in $x + y + z$
4. Compute clause conflict
5. Propagate

	$\mathcal{D}$	*reason*
x	$]-\infty;+\infty[$	
y	$]-\infty;+\infty[$	
z	$]-\infty;0]$	...
x-y	$[10;+\infty[$	
x+y+z	$]10;+\infty[$	

Example: Arithmetic

$$10 \leq x - y \quad \wedge \quad 10 < x + y + z$$

1. Decide $x = 0$
2. Propagate using $x - y$
3. Conflict in $x + y + z$
4. Compute clause conflict
5. Propagate

	$\mathcal{D}$	*reason*
x	$\{0\}$	*dec*
y	$]-\infty;+\infty[$	
z	$]-\infty;0]$	...
x-y	$[10;+\infty[$	
x+y+z	$]10;+\infty[$	

Example: Arithmetic

$$10 \leq x - y \quad \wedge \quad 10 < x + y + z$$

1. Decide $x = 0$
2. Propagate using $x - y$
3. Conflict in $x + y + z$
4. Compute clause conflict
5. Propagate

	$\mathcal{D}$	*reason*
x	$\{0\}$	*dec*
y	$]-\infty; -10]$	$x - y$
z	$]-\infty; 0]$	...
x-y	$[10; +\infty[$	
x+y+z	$]10; +\infty[$	

Example: Arithmetic

$$10 \leq x - y \quad \wedge \quad 10 < x + y + z$$

1. Decide $x = 0$
2. Propagate using $x - y$
3. Conflict in $x + y + z$
4. Compute clause conflict
5. Propagate

	$\mathcal{D}$	*reason*
x	$\{0\}$	*dec*
y	$]-\infty; -10]$	$x - y$
z	$]-\infty; 0]$	...
x-y	$[10; +\infty[$	
x+y+z	$]10; +\infty[$	

Example: Arithmetic

$$10 \leq x - y \quad \wedge \quad 10 < x + y + z$$

1. Decide $x = 0$
2. Propagate using $x - y$
3. Conflict in $x + y + z$
4. Compute clause conflict
5. Propagate

	$\mathcal{D}$	*reason*
x	$\{0\}$	*dec*
y	$]-\infty; -10]$	$x - y$
z	$]-\infty; 0]$	...
x-y	$[10; +\infty[$	
x+y+z	$]10; +\infty[$	

$$\dfrac{\dfrac{\dfrac{\texttt{dec}}{0 \leq -x} \quad \dfrac{\texttt{level0}}{10 \leq x - y}}{10 \leq -y} \quad \dfrac{\texttt{dec}}{0 \leq -x} \quad \dfrac{\vdots}{0 \leq -z} \quad \dfrac{\texttt{level0}}{10 < x + y + z}}{20 < 0}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{-x \leq -x}\,\texttt{gen} \quad \dfrac{}{10 \leq x - y}\,\texttt{level0}}{-x + 10 \leq -y} \quad \dfrac{\dfrac{}{-x \leq -x}\,\texttt{gen} \quad \dfrac{}{-z \leq -z}\,\texttt{gen} \quad \dfrac{}{10 < x + y + }\,\texttt{level0}}{}}{-z - 2x + 20 < 0}$$

Example: Arithmetic

$$10 \leq x - y \quad \wedge \quad 10 < x + y + z$$

1. Decide $x = 0$
2. Propagate using $x - y$
3. Conflict in $x + y + z$
4. Compute clause conflict
5. Propagate

	$\mathcal{D}$	*reason*
x	$\{0\}$	*dec*
y	$]-\infty; -10]$	$x - y$
z	$]-\infty; 0]$	...
x-y	$[10; +\infty[$	
x+y+z	$]10; +\infty[$	

Example: Arithmetic

$$10 \leq x - y \quad \wedge \quad 10 < x + y + z$$

1. Decide $x = 0$
2. Propagate using $x - y$
3. Conflict in $x + y + z$
4. Compute clause conflict
5. Propagate

	$\mathcal{D}$	*reason*
x	$\{0\}$	*dec*
y	$]-\infty; -10]$	$x - y$
z	$]-\infty; 0]$	...
x-y	$[10; +\infty[$	
x+y+z	$]10; +\infty[$	
2x+z	$[20; +\infty[$	

Example: Arithmetic

$$10 \leq x - y \quad \wedge \quad 10 < x + y + z$$

1. Decide $x = 0$
2. Propagate using $x - y$
3. Conflict in $x + y + z$
4. Compute clause conflict
5. Propagate

	$\mathcal{D}$	*reason*
x	$]10; +\infty]$	$2x + z$
y	$]-\infty; +\infty[$	
z	$]-\infty; 0]$	...
x-y	$[10; +\infty[$	
x+y+z	$]10; +\infty[$	
2x+z	$[20; +\infty[$	

Learning(SMT) and Propagation(CP)

- Design new theory decision procedures that use domains, decisions and learning at there heart.
- Easy combination during propagation
- Combination during learning using a common language for each domains

SOPRANO

New framework:

- First class domains
- First class learning
- No special case (ex: booleans, uninterpreted function)
- Easy model generation
- Easy non-convexe theory

Theories:

- linear, non-linear arithmetic
- abstraction of arrays value
- floating-points
- quantifiers
- bitvectors, algebraic datatypes

SOPRANO

New framework:

- First class domains
- First class learning
- No special case (ex: booleans, uninterpreted function)
- Easy model generation
- Easy non-convexe theory

Theories:

- linear, non-linear arithmetic
- abstraction of arrays value
- floating-points
- quantifiers
- bitvectors, algebraic datatypes

SOPRANO

New framework:

- First class domains
- First class learning
- No special case (ex: booleans, uninterpreted function)
- Easy model generation
- Easy non-convexe theory

Theories:

- linear, non-linear arithmetic
- abstraction of arrays value
- floating-points
- quantifiers
- bitvectors, algebraic datatypes

SOPRANO

New framework:

- First class domains
- First class learning
- No special case (ex: booleans, uninterpreted function)
- Easy model generation
- Easy non-convexe theory

Theories:

- linear, non-linear arithmetic
- abstraction of arrays value
- floating-points
- quantifiers
- bitvectors, algebraic datatypes

SOPRANO

New framework:

- First class domains
- First class learning
- No special case (ex: booleans, uninterpreted function)
- Easy model generation
- Easy non-convexe theory

Theories:

- linear, non-linear arithmetic
- abstraction of arrays value
- floating-points
- quantifiers
- bitvectors, algebraic datatypes

SOPRANO

New framework:

- First class domains
- First class learning
- No special case (ex: booleans, uninterpreted function)
- Easy model generation
- Easy non-convexe theory

Theories:

- linear, non-linear arithmetic
- abstraction of arrays value
- floating-points
- quantifiers
- bitvectors, algebraic datatypes

SOPRANO

New framework:

- First class domains
- First class learning
- No special case (ex: booleans, uninterpreted function)
- Easy model generation
- Easy non-convexe theory

Theories:

- linear, non-linear arithmetic
- abstraction of arrays value
- floating-points
- quantifiers
- bitvectors, algebraic datatypes

SOPRANO

New framework:

- First class domains
- First class learning
- No special case (ex: booleans, uninterpreted function)
- Easy model generation
- Easy non-convexe theory

Theories:

- linear, non-linear arithmetic
- abstraction of arrays value
- floating-points
- quantifiers
- bitvectors, algebraic datatypes

SOPRANO

New framework:

- First class domains
- First class learning
- No special case (ex: booleans, uninterpreted function)
- Easy model generation
- Easy non-convexe theory

Theories:

- linear, non-linear arithmetic
- abstraction of arrays value
- floating-points
- quantifiers
- bitvectors, algebraic datatypes

SOPRANO

New framework:

- First class domains
- First class learning
- No special case (ex: booleans, uninterpreted function)
- Easy model generation
- Easy non-convexe theory

Theories:

- linear, non-linear arithmetic
- abstraction of arrays value
- floating-points
- quantifiers
- bitvectors, algebraic datatypes