Recent Trends in QBF Solving

Friedrich Slivovsky

Algorithms & Complexity Group, TU Wien

Quantified Boolean Fundamentals

SAT vs. QBF

SAT

$$\varphi(x_1,\ldots,x_n)$$

SAT vs. QBF

SAT

$$\exists x_1 \exists x_2 \exists x_3 \exists x_4 \ \ldots \ \exists x_{n-1} \exists x_n \ . \ \varphi(x_1, \ldots, x_n)$$

SAT vs. QBF

SAT

$$\exists x_1 \exists x_2 \exists x_3 \exists x_4 \ \ldots \ \exists x_{n-1} \exists x_n \ . \ \varphi(x_1, \ldots, x_n)$$

QBF

$$\forall x_1 \exists x_2 \forall x_3 \exists x_4 \ \ldots \ \forall x_{n-1} \exists x_n \ . \ \varphi(x_1, \ldots, x_n)$$

SAT vs. QBF

SAT

SAT vs. QBF

SAT **NP complete**

SAT vs. QBF

SAT **NP complete**

but CDCL SAT solvers are very efficient

SAT vs. QBF

SAT **NP complete**

but CDCL SAT solvers are very efficient

QBF

SAT vs. QBF

SAT **NP complete**

but CDCL SAT solvers are very efficient

QBF **PSPACE complete**

SAT vs. QBF

SAT **NP complete**

but CDCL SAT solvers are very efficient

QBF **PSPACE complete**

but QBF encodings can be much more succinct

SAT vs. QBF

SAT vs. QBF

Faymonville et. al.: **Encodings of Bounded Synthesis**. TACAS 2017

SAT vs. QBF

Faymonville et. al.: **Encodings of Bounded Synthesis**. TACAS 2017

SAT

$O(nm^2 \mathbf{2^l} (d + n \log(nm)))$

SAT vs. QBF

Faymonville et. al.: **Encodings of Bounded Synthesis**. TACAS 2017

SAT	**QBF**
$O(nm^2 \mathbf{2^l} (d + n \log(nm)))$	$O(nm^2 (d + n \log(nm)))$

SAT vs. QBF

Faymonville et. al.: **Encodings of Bounded Synthesis**. TACAS 2017

SAT	**QBF**	**DQBF**
$O(nm^2 \mathbf{2^l} (d + n \log(nm)))$	$O(nm^2 (d + n \log(nm)))$	$O(n+m+d+\log(nm))$

SAT vs. QBF

Faymonville et. al.: **Encodings of Bounded Synthesis**. TACAS 2017

SAT	**QBF**	**DQBF**
$O(nm^2 \mathbf{2^l} (d + n \log(nm)))$	$O(nm^2 (d + n \log(nm)))$	$O(n+m+d+\log(nm))$

QBF encoding significantly better than **SAT** and **DQBF**

Semantics (Assignment)

Semantics (Assignment)

```
AssignQBF(Qx ϕ):
```

Semantics (Assignment)

```
AssignQBF(Qx ϕ):
  if ϕ is trivially true:
```

Semantics (Assignment)

```
AssignQBF(Qx ϕ):
  if ϕ is trivially true:
    return True
```

Semantics (Assignment)

```
AssignQBF(Qx ϕ):
  if ϕ is trivially true:
    return True
  else if ϕ is trivially false:
```

Semantics (Assignment)

```
AssignQBF(Qx ϕ):
  if ϕ is trivially true:
    return True
  else if ϕ is trivially false:
    return False
```

Semantics (Assignment)

```
AssignQBF(Qx ϕ):
  if ϕ is trivially true:
    return True
  else if ϕ is trivially false:
    return False
  else if Q == ∃:
```

Semantics (Assignment)

```
AssignQBF(Qx ϕ):
  if ϕ is trivially true:
    return True
  else if ϕ is trivially false:
    return False
  else if Q == ∃:
    return AssignQBF(ϕ[x=F]) || AssignQBF(ϕ[x=T])
```

Semantics (Assignment)

```
AssignQBF(Qx ϕ):
  if ϕ is trivially true:
    return True
  else if ϕ is trivially false:
    return False
  else if Q == ∃:
    return AssignQBF(ϕ[x=F]) || AssignQBF(ϕ[x=T])
  else:
```

Semantics (Assignment)

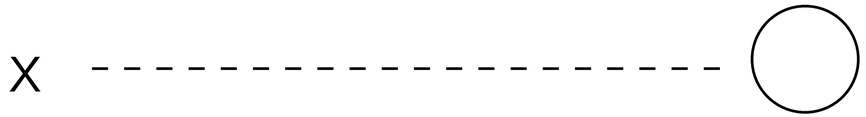
```
AssignQBF(Qx ϕ):
  if ϕ is trivially true:
    return True
  else if ϕ is trivially false:
    return False
  else if Q == ∃:
    return AssignQBF(ϕ[x=F]) || AssignQBF(ϕ[x=T])
  else:
    return AssignQBF(ϕ[x=F]) && AssignQBF(ϕ[x=T])
```

Semantics (Assignment)

$$\forall x \, \exists y \, . \, (x \vee \neg y) \wedge (\neg x \vee y)$$

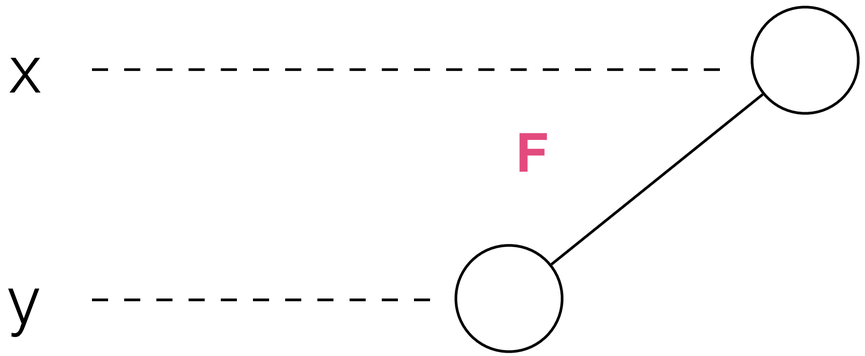
Semantics (Assignment)

$$\forall x \, \exists y \, . \, (x \vee \neg y) \wedge (\neg x \vee y)$$



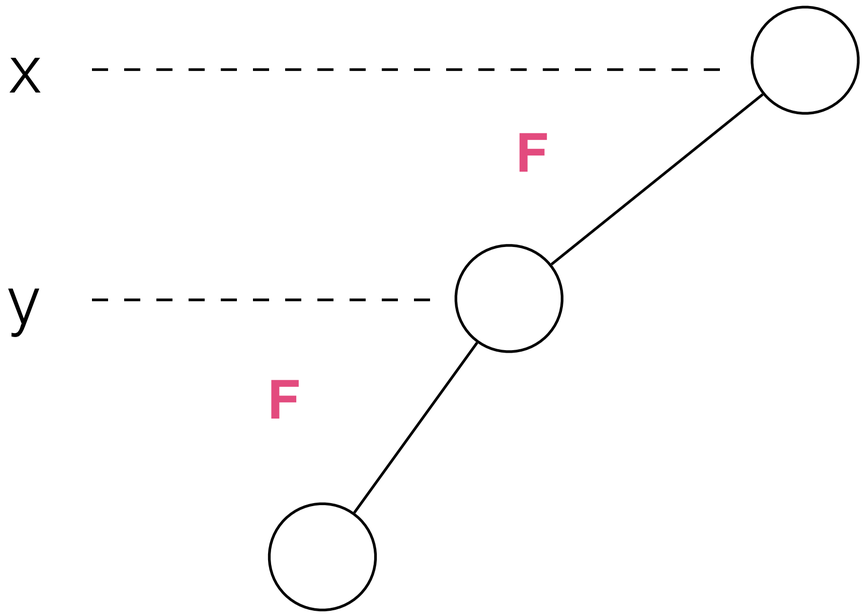
Semantics (Assignment)

$$\forall x \, \exists y \, . \, (x \vee \neg y) \wedge (\neg x \vee y)$$



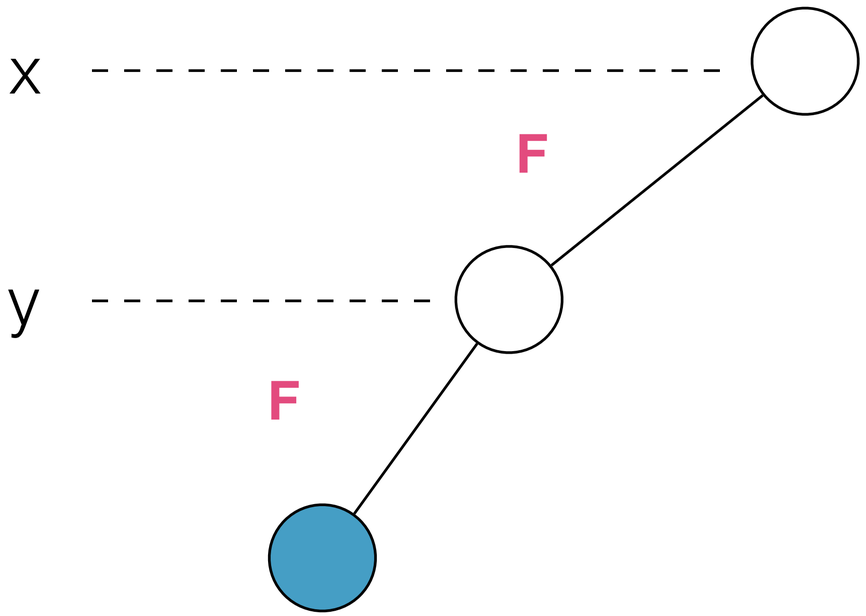
Semantics (Assignment)

$$\forall x \, \exists y \, . \, (x \vee \neg y) \wedge (\neg x \vee y)$$



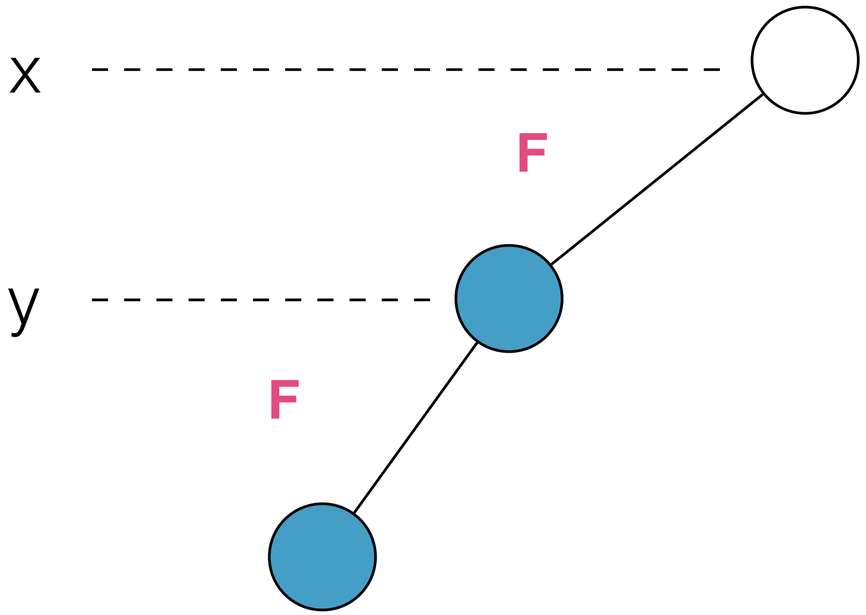
Semantics (Assignment)

$$\forall x \, \exists y \, . \, (x \vee \neg y) \wedge (\neg x \vee y)$$



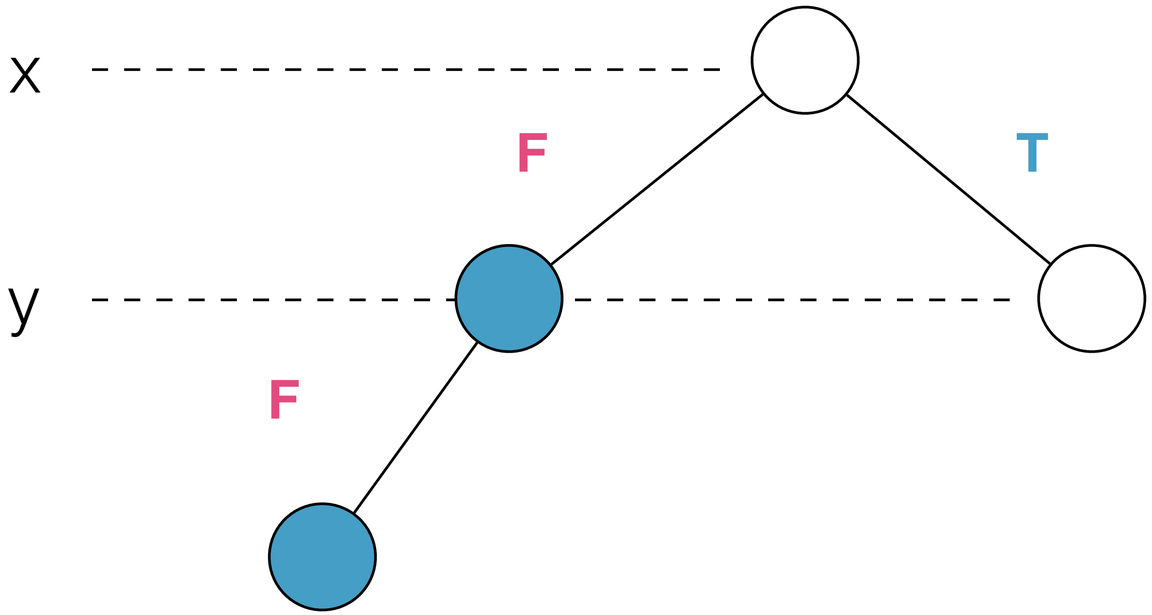
Semantics (Assignment)

$$\forall x \, \exists y \, . \, (x \vee \neg y) \wedge (\neg x \vee y)$$



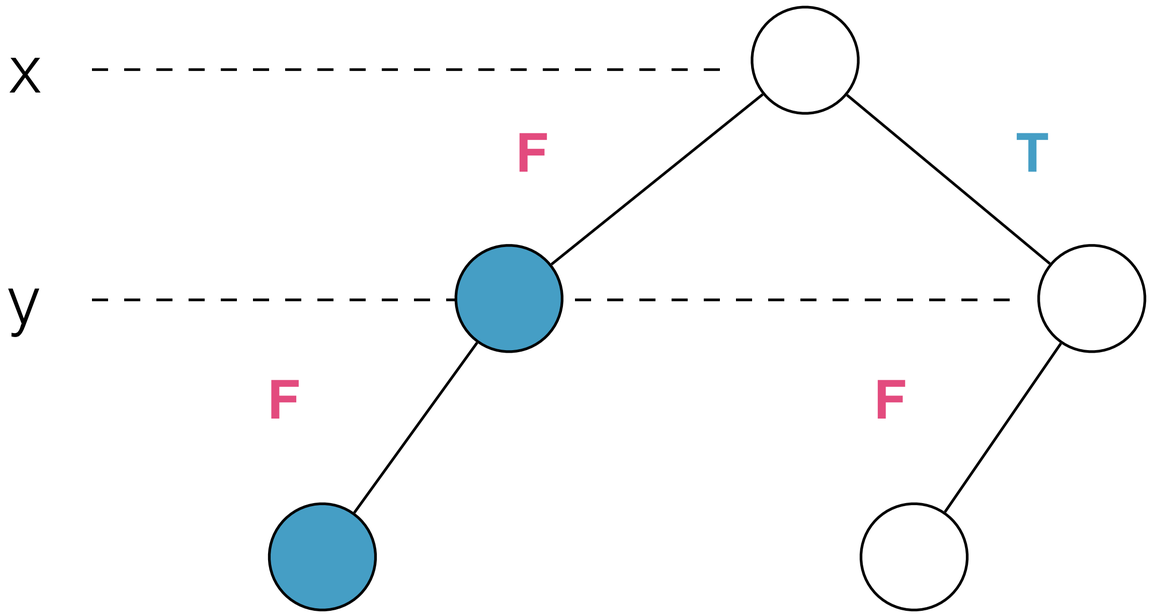
Semantics (Assignment)

$$\forall x \, \exists y \, . \, (x \vee \neg y) \wedge (\neg x \vee y)$$



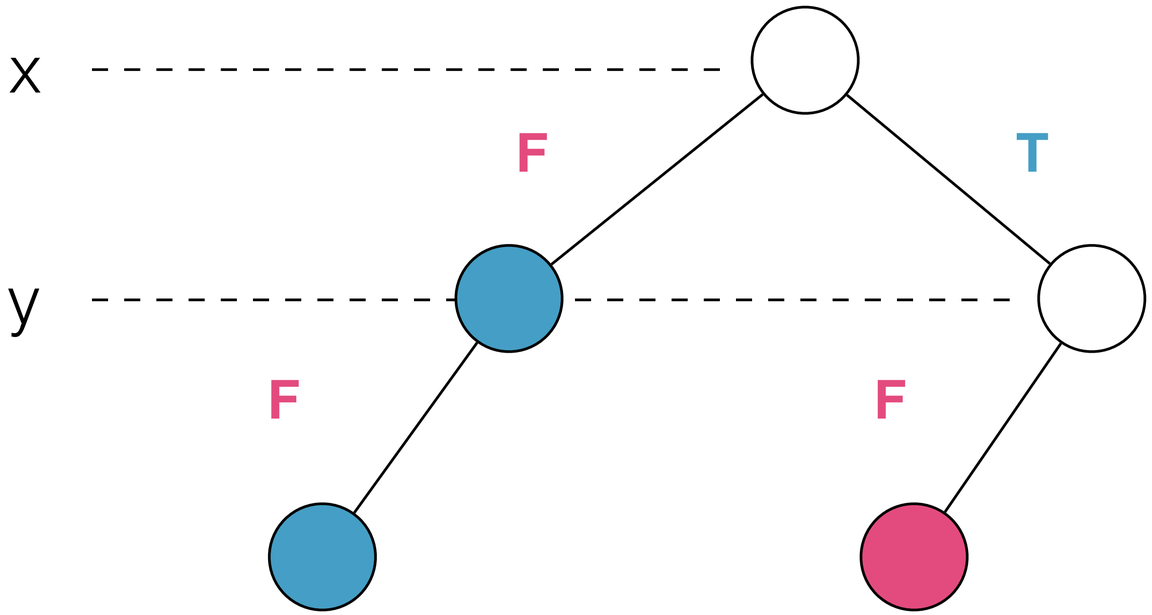
Semantics (Assignment)

$$\forall x \, \exists y \, . \, (x \vee \neg y) \wedge (\neg x \vee y)$$



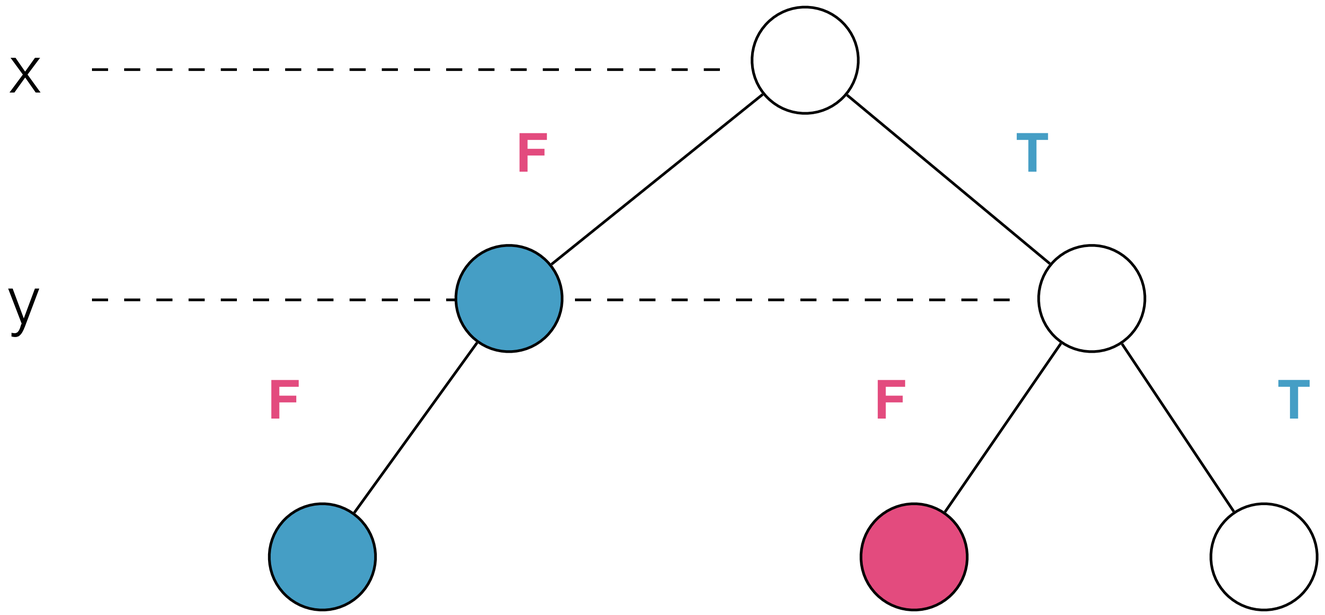
Semantics (Assignment)

$$\forall x \, \exists y \, . \, (x \vee \neg y) \wedge (\neg x \vee y)$$



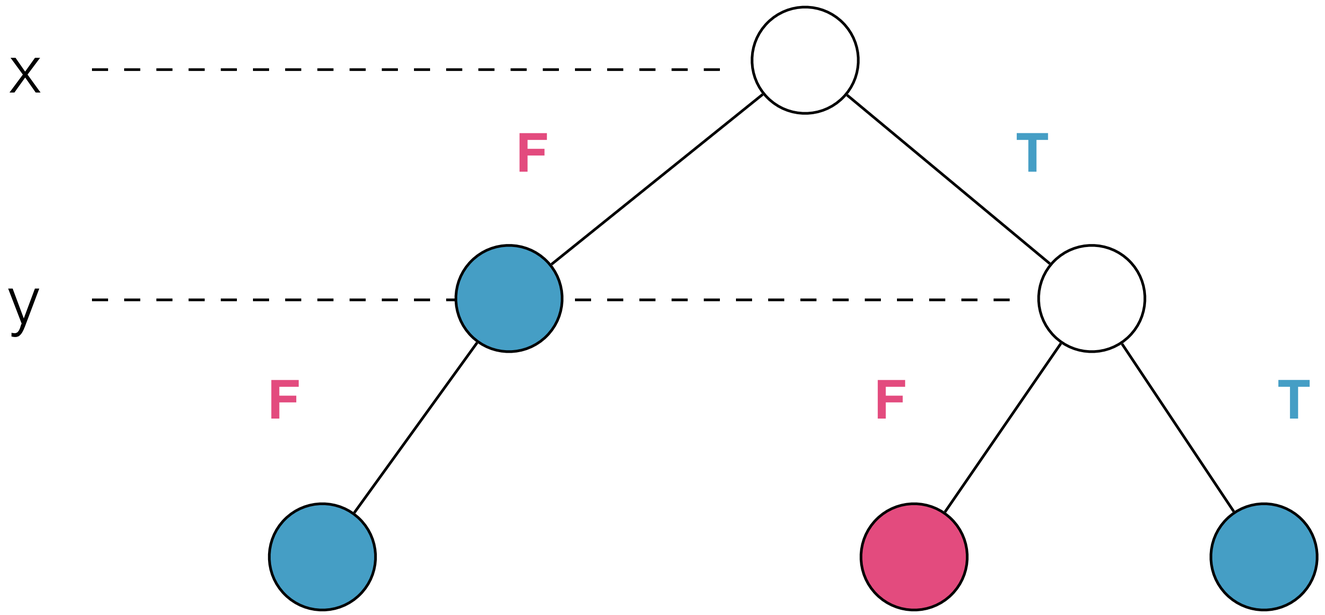
Semantics (Assignment)

$$\forall x \, \exists y \, . \, (x \vee \neg y) \wedge (\neg x \vee y)$$



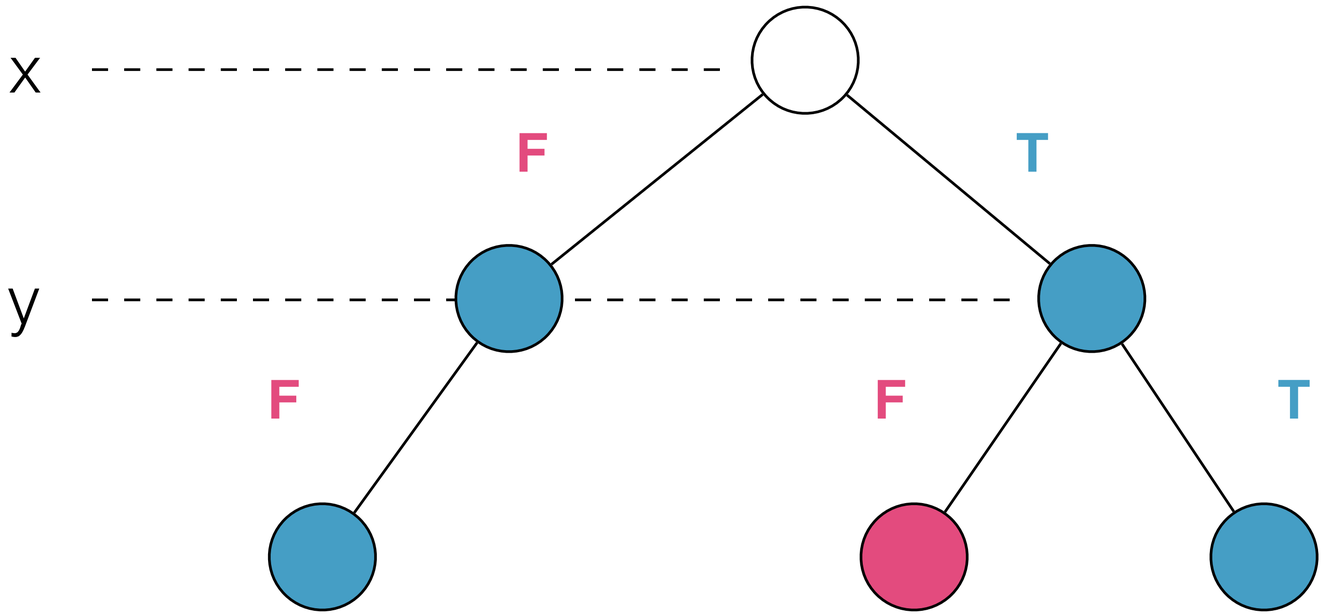
Semantics (Assignment)

$$\forall x \, \exists y \, . \, (x \vee \neg y) \wedge (\neg x \vee y)$$



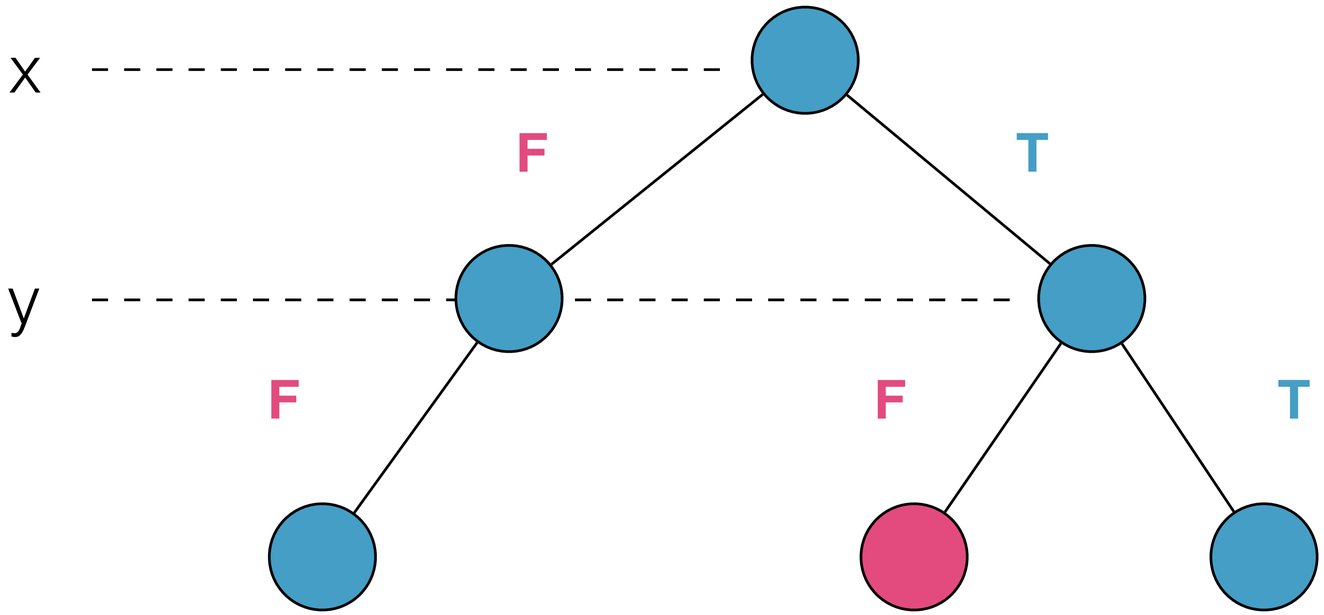
Semantics (Assignment)

$$\forall x \, \exists y \, . \, (x \vee \neg y) \wedge (\neg x \vee y)$$



Semantics (Assignment)

$$\forall x \, \exists y \, . \, (x \vee \neg y) \wedge (\neg x \vee y)$$



Semantics (Expansion)

$\forall x\, \phi(x)$

Semantics (Expansion)

$$\forall x\ \phi(x) \equiv \phi(\mathbf{T}) \wedge \phi(\mathbf{F})$$

Semantics (Expansion)

$$\forall x\ \phi(x) \equiv \phi(\mathbf{T}) \wedge \phi(\mathbf{F})$$

```
ExpandQBF(ϕ):
 while ϕ contains universally quantified x:
   ϕ = Prenex(Expand(x, ϕ))
 return SAT(ϕ)
```

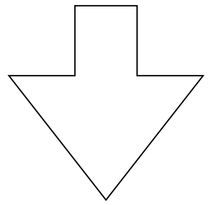
Semantics (Expansion)

Semantics (Expansion)

$\forall x_1 \exists y \forall x_2 \exists z \,.\, \varphi(x_1, x_2, y, z)$

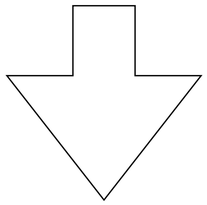
Semantics (Expansion)

$\forall x_1 \exists y \forall x_2 \exists z \,.\, \varphi(x_1, x_2, y, z)$



Semantics (Expansion)

$$\forall x_1 \exists y \forall x_2 \exists z \;.\; \varphi(x_1, x_2, y, z)$$



$$\exists y_{\mathbf{F}}, z_{\mathbf{FF}}, z_{\mathbf{FT}}, y_{\mathbf{T}}, z_{\mathbf{TF}}, z_{\mathbf{TT}} \;.\; \varphi(\mathbf{F}, \mathbf{F}, y_{\mathbf{F}}, z_{\mathbf{FF}}) \wedge \varphi(\mathbf{F}, \mathbf{T}, y_{\mathbf{F}}, z_{\mathbf{FT}}) \wedge \varphi(\mathbf{T}, \mathbf{F}, y_{\mathbf{T}}, z_{\mathbf{TF}}) \wedge \varphi(\mathbf{T}, \mathbf{T}, y_{\mathbf{T}}, z_{\mathbf{TT}})$$

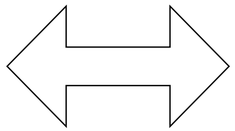
Semantics (Strategies)

Semantics (Strategies)

$\forall x \exists y \forall w \exists z \; . \; \varphi(x,w,y,z)$

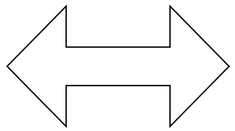
Semantics (Strategies)

$$\forall x \exists y \forall w \exists z \;.\; \varphi(x,w,y,z)$$

True 

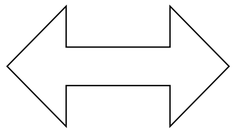
Semantics (Strategies)

$$\forall x \exists y \forall w \exists z \;.\; \varphi(x,w,y,z)$$

True  $f_y(x)$ $f_z(x,w)$

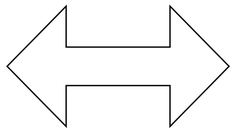
Semantics (Strategies)

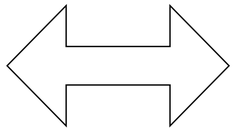
$$\forall x \exists y \forall w \exists z \;.\; \varphi(x,w,y,z)$$

True  $f_y(x)$ $f_z(x,w)$ $\varphi(x,f_y(x),w,f_z(x,w))$ **valid**

Semantics (Strategies)

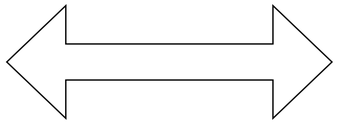
$$\forall x \exists y \forall w \exists z \;.\; \varphi(x,w,y,z)$$

True  $f_y(x)\ \ f_z(x,w)$ $\varphi(x,f_y(x),w,f_z(x,w))$ **valid**

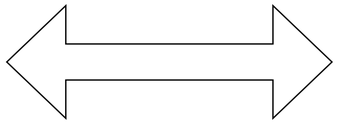
False  $f_x()\ \ f_w(y)$ $\varphi(f_x(),y,f_w(y),z)$ **unsat**

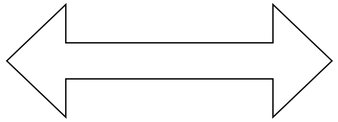
Solver Paradigms

Solver Paradigms

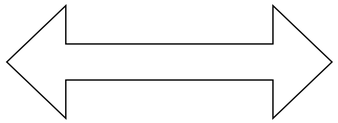
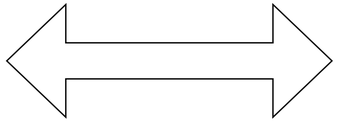
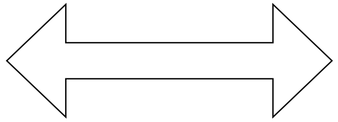
Assignment  QCDCL

Solver Paradigms

Assignment  QCDCL

Expansion  CEGAR

Solver Paradigms

Assignment		QCDCL
Expansion		CEGAR
Strategies		Certification/?

Commercial Break QBF Tools

Workflow(s)

Workflow(s)

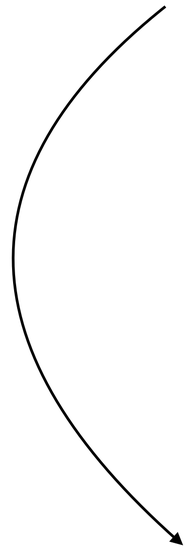
QDIMACS

Prenex CNF

Workflow(s)

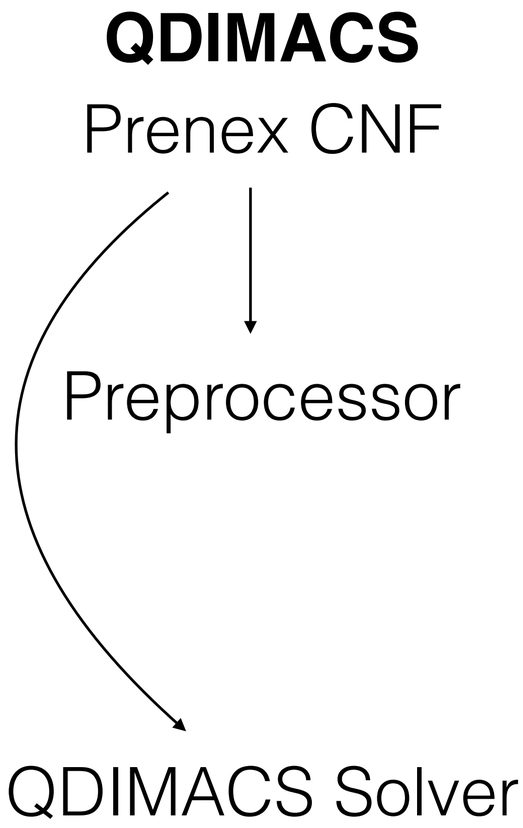
QDIMACS

Prenex CNF

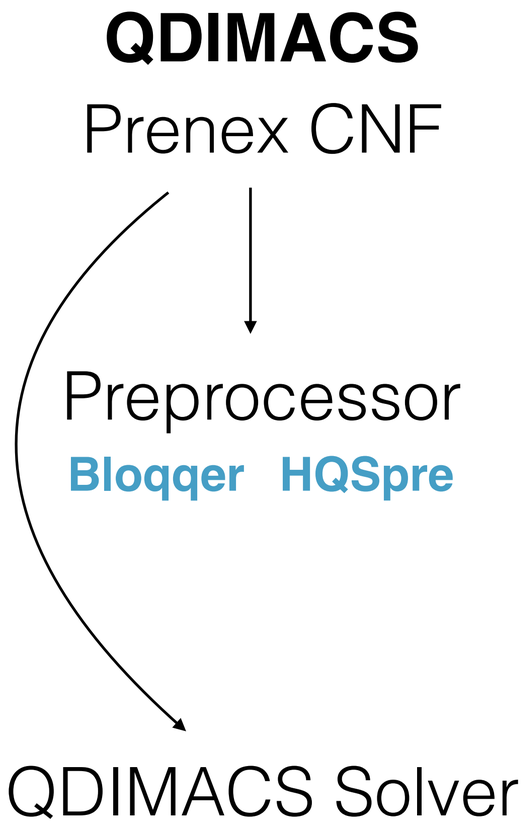


QDIMACS Solver

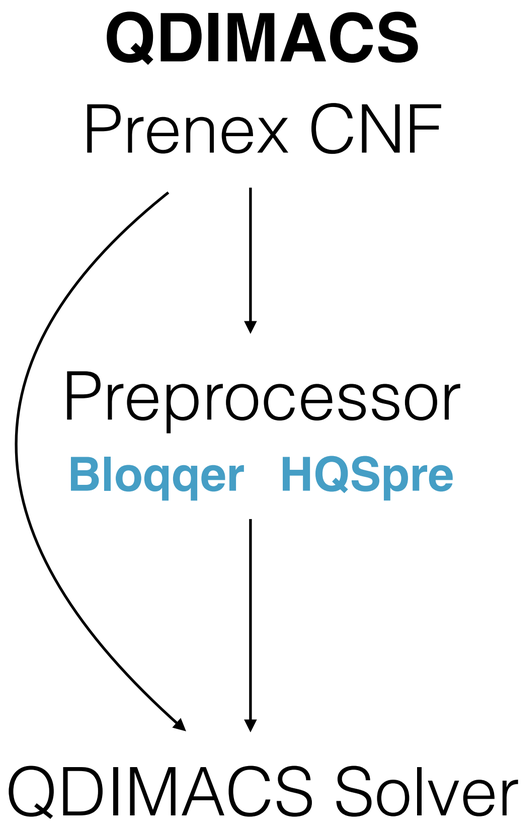
Workflow(s)



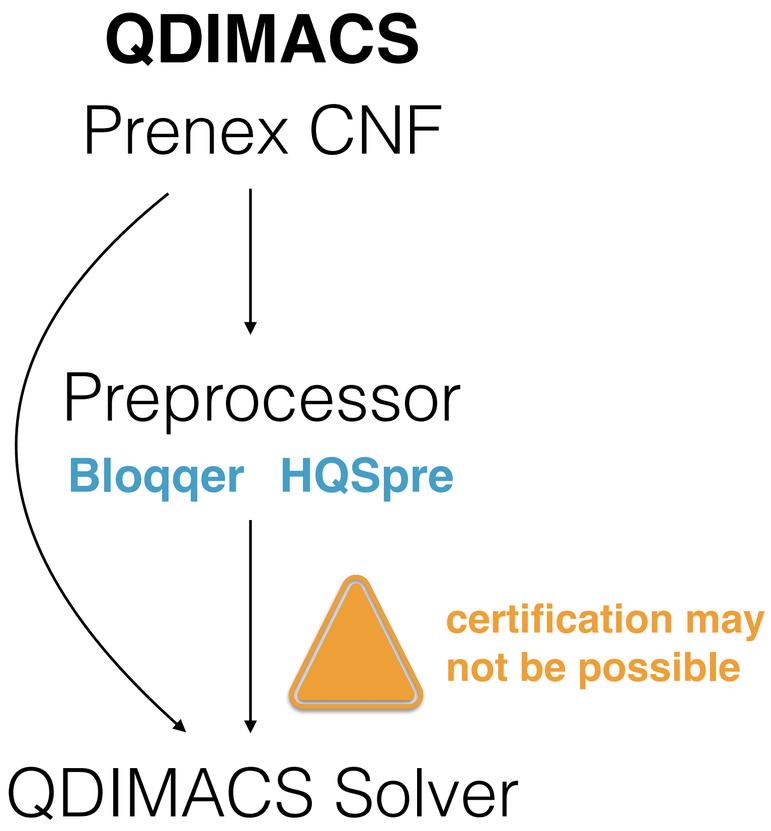
Workflow(s)


QDIMACS
Prenex CNF
Preprocessor
Bloqqer HQSpre
QDIMACS Solver

Workflow(s)


QDIMACS
Prenex CNF
Preprocessor
Bloqqer HQSpre
QDIMACS Solver

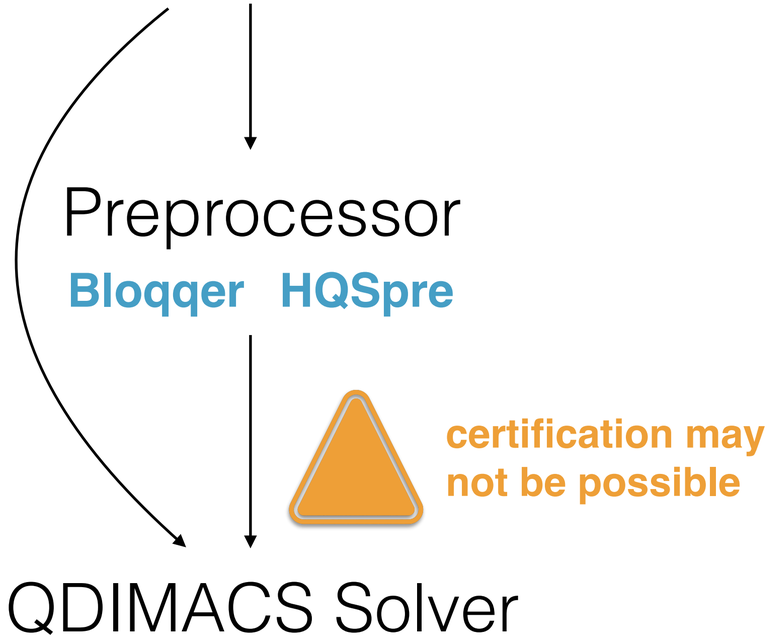
Workflow(s)


QDIMACS
Prenex CNF
Preprocessor
Bloqqer HQSpre
certification may not be possible
QDIMACS Solver

Workflow(s)

QDIMACS

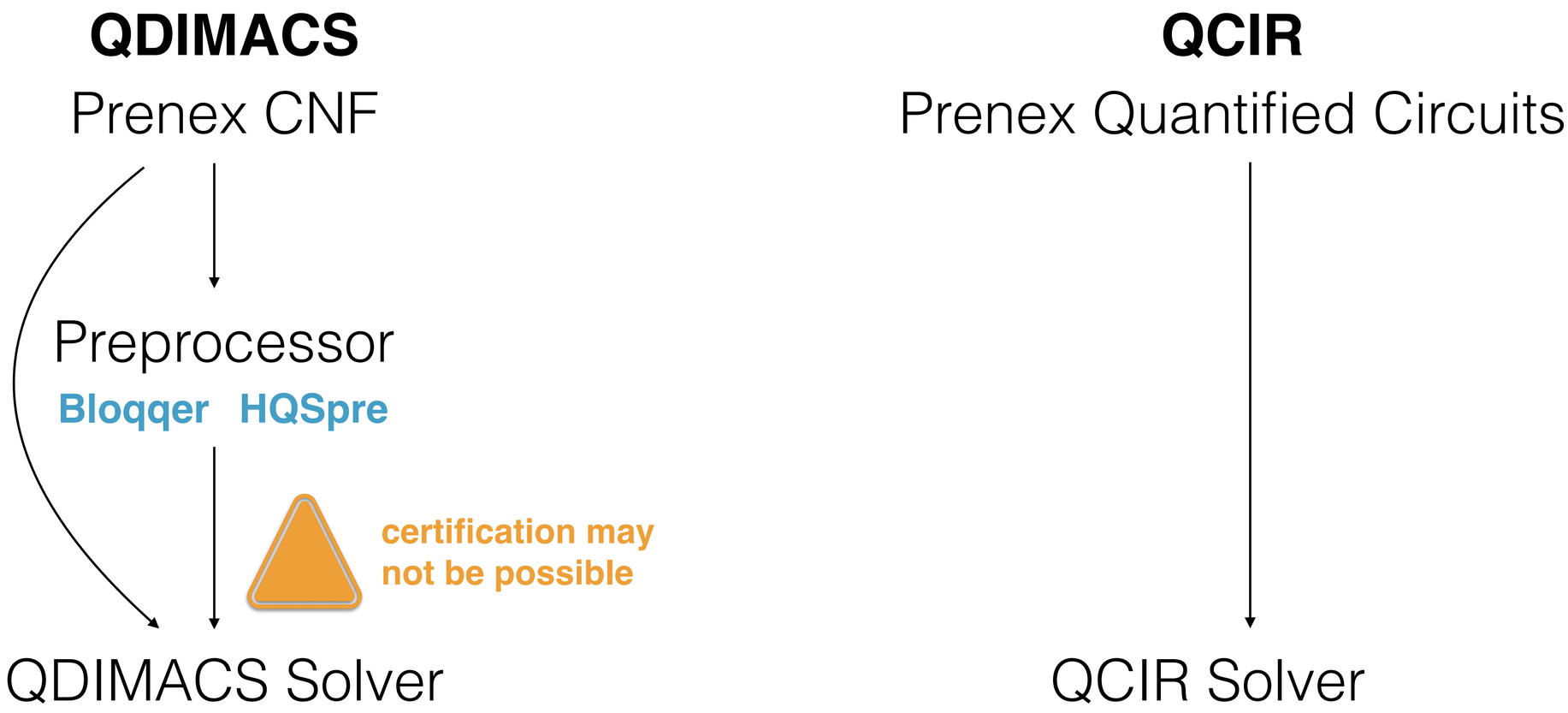
Prenex CNF



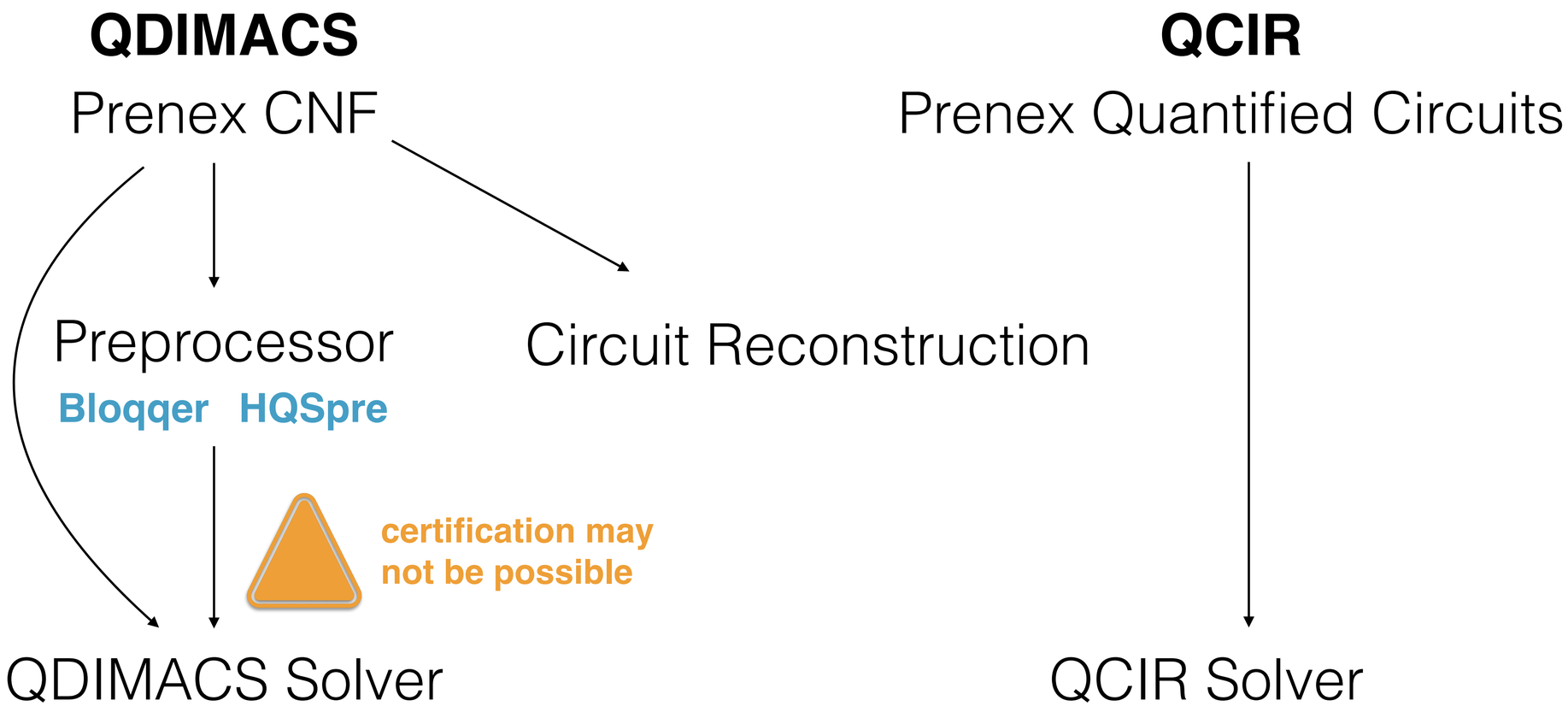
QCIR

Prenex Quantified Circuits

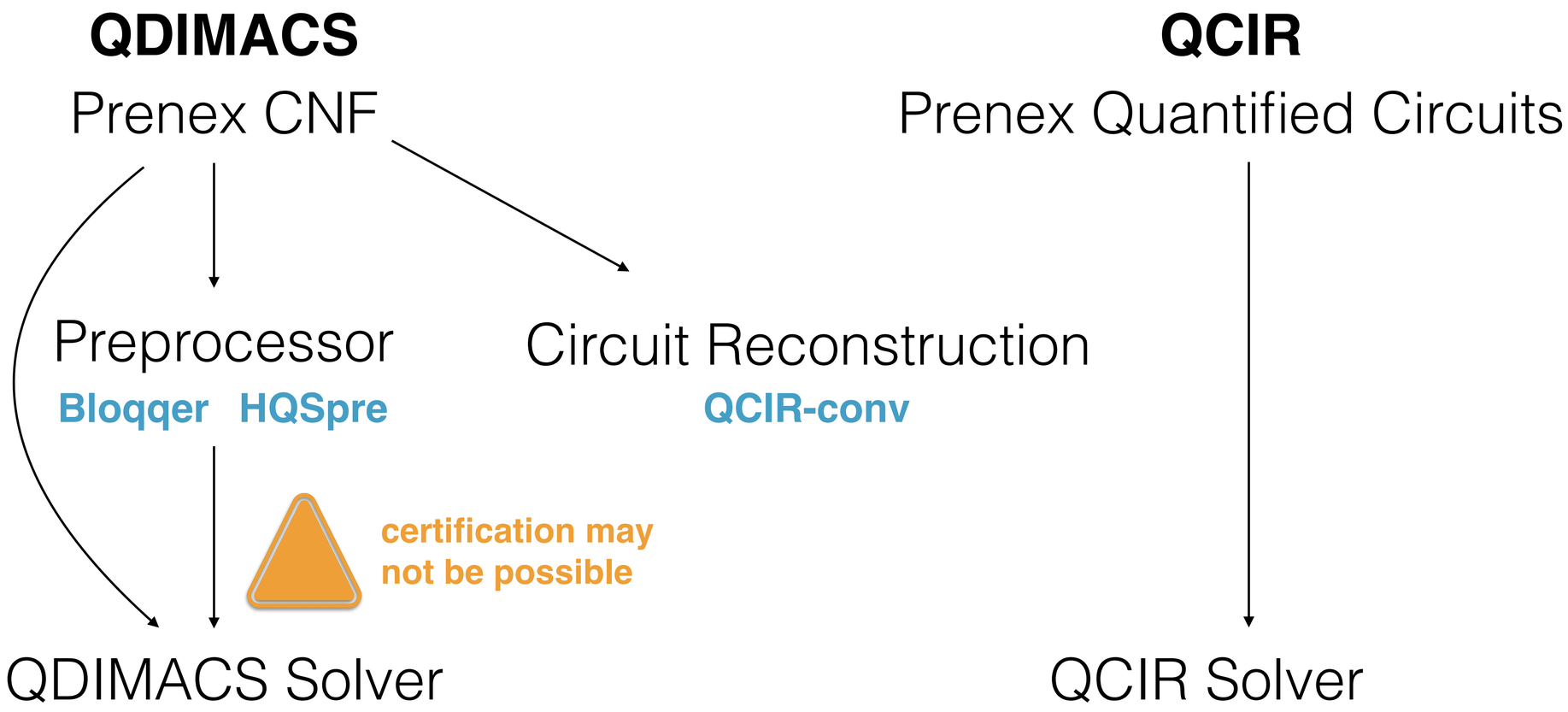
Workflow(s)


QDIMACS
Prenex CNF
Preprocessor
Bloqqer HQSpre
certification may not be possible
QDIMACS Solver
QCIR
Prenex Quantified Circuits
QCIR Solver

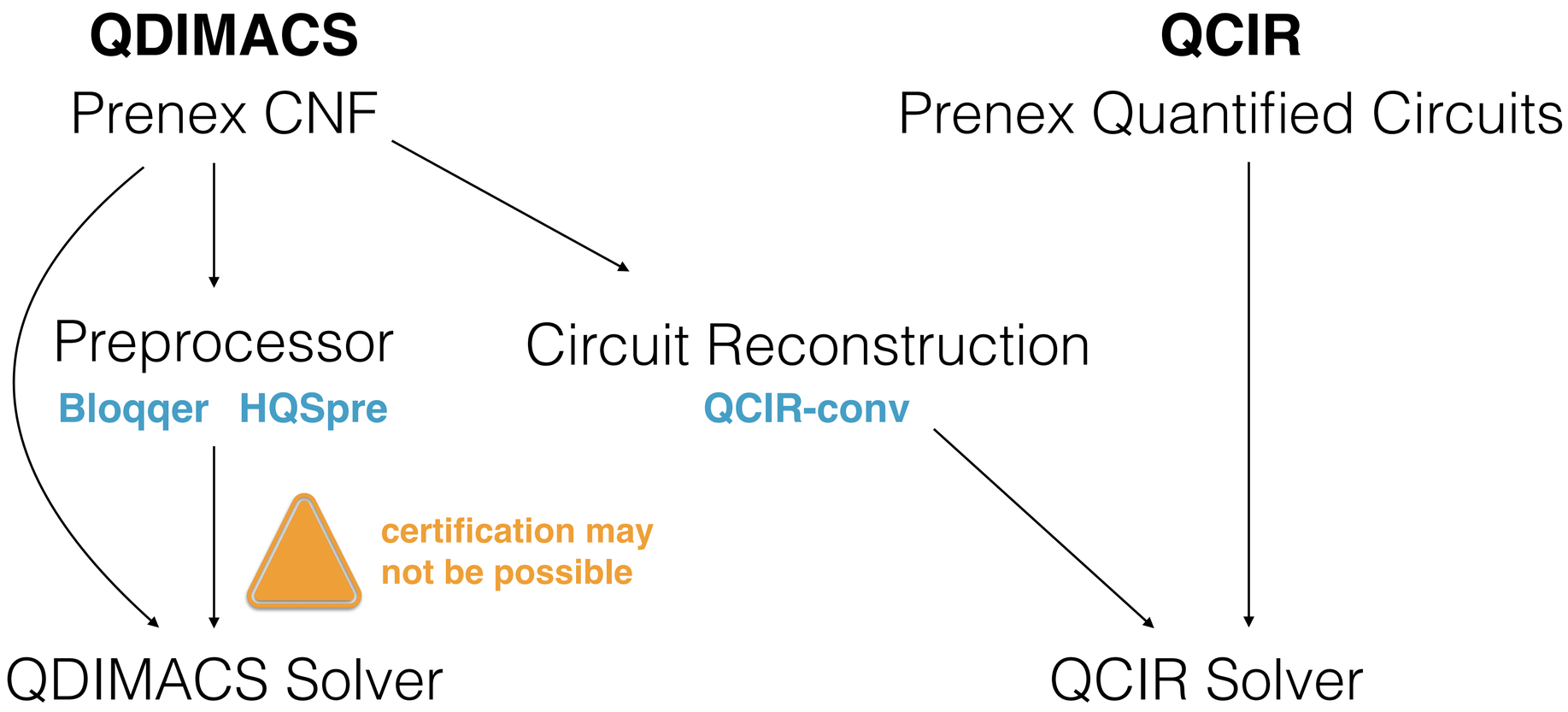
Workflow(s)


QDIMACS
Prenex CNF
QCIR
Prenex Quantified Circuits
Preprocessor
Bloqqer HQSpre
Circuit Reconstruction
certification may
not be possible
QDIMACS Solver
QCIR Solver

Workflow(s)


QDIMACS
Prenex CNF
QCIR
Prenex Quantified Circuits
Preprocessor
Bloqqer HQSpre
Circuit Reconstruction
QCIR-conv
certification may
not be possible
QDIMACS Solver
QCIR Solver

Workflow(s)


QDIMACS
Prenex CNF
QCIR
Prenex Quantified Circuits
Preprocessor
Bloqqer HQSpre
Circuit Reconstruction
QCIR-conv
certification may
not be possible
QDIMACS Solver
QCIR Solver

Solvers

QDIMACS

DepQBF
Qute
CAQE
RAReQS
Qesto
GhostQ

QCIR

Qute
Quabs
GhostQ
QFun

Solvers

QDIMACS

DepQBF
Qute
CAQE
RAReQS
Qesto
GhostQ

QCIR

Qute
Quabs
GhostQ
QFun

see QBFEval website

Recent Developments

Q: CDCL works very well for SAT, why doesn’t it work as well for QBF?

CDCL

CDCL

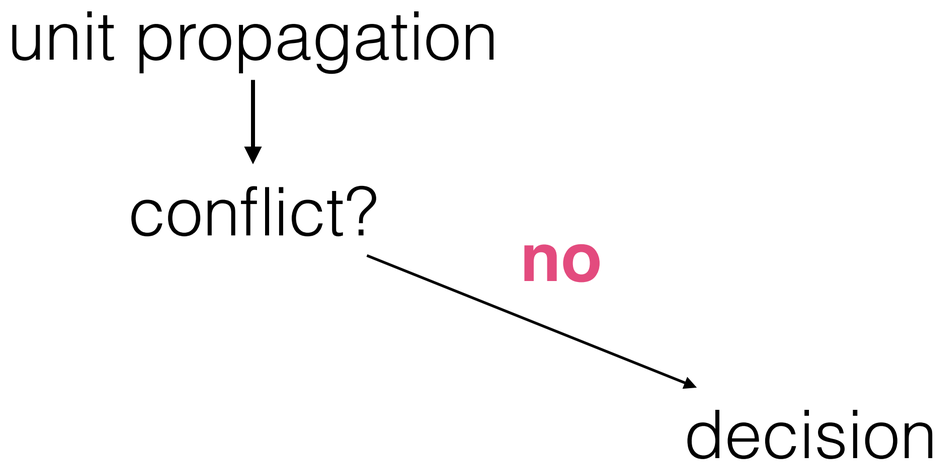
unit propagation

CDCL

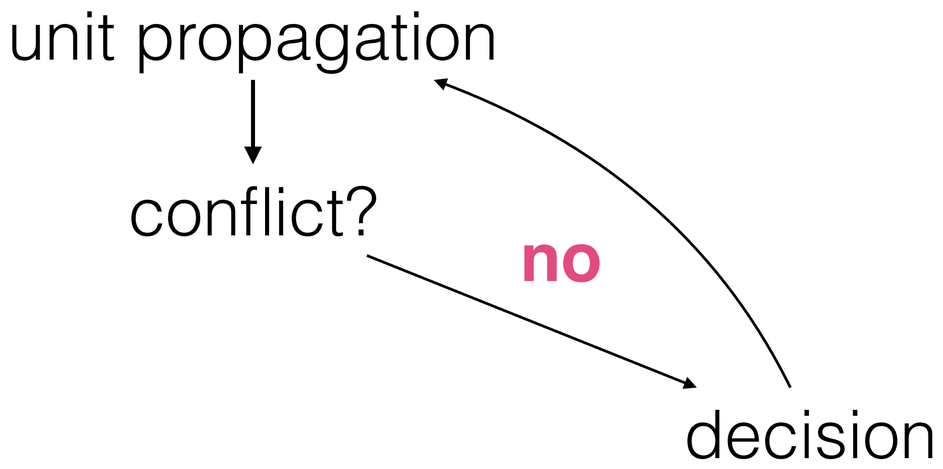
unit propagation

conflict?

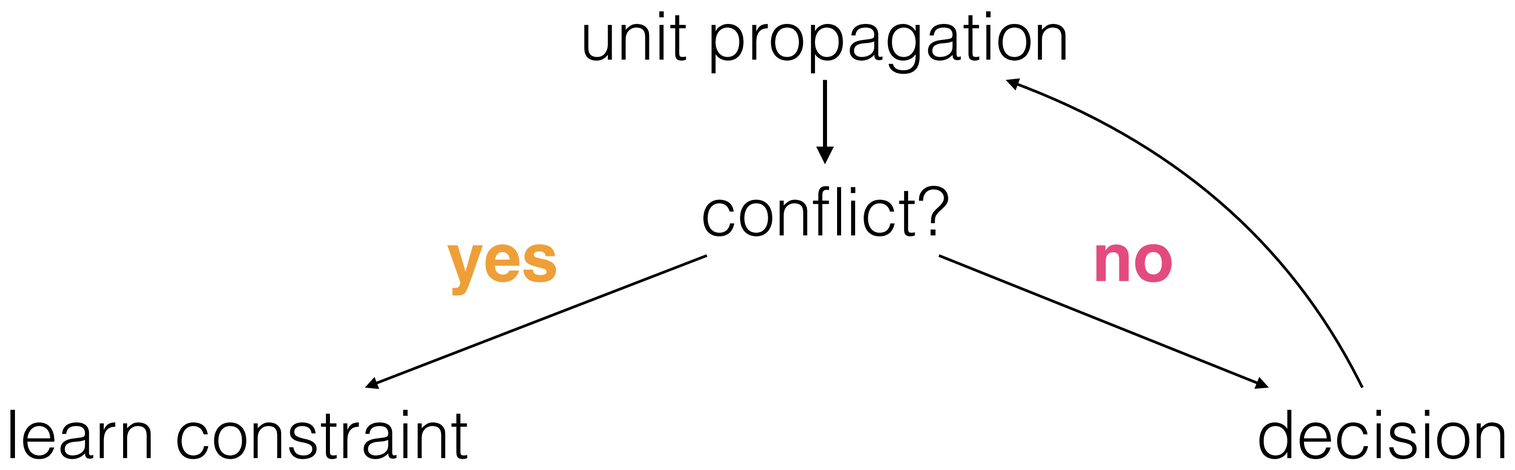
CDCL



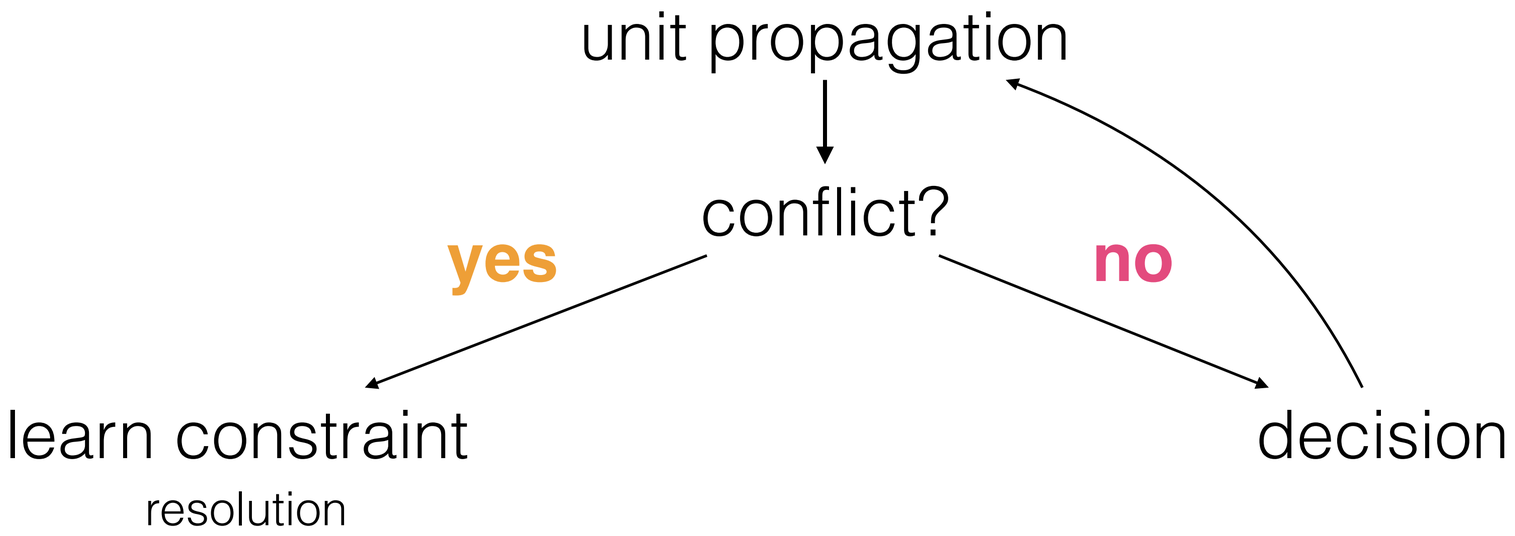
CDCL



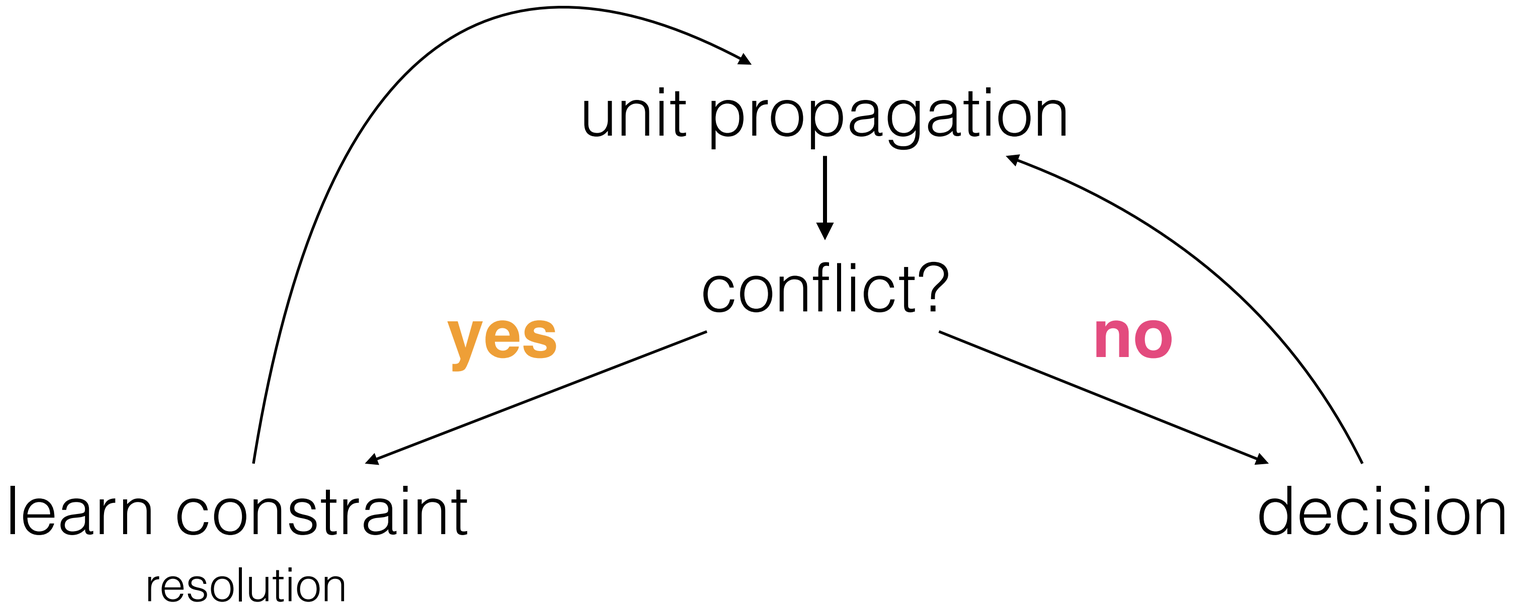
CDCL


unit propagation
conflict?
yes
no
learn constraint
decision

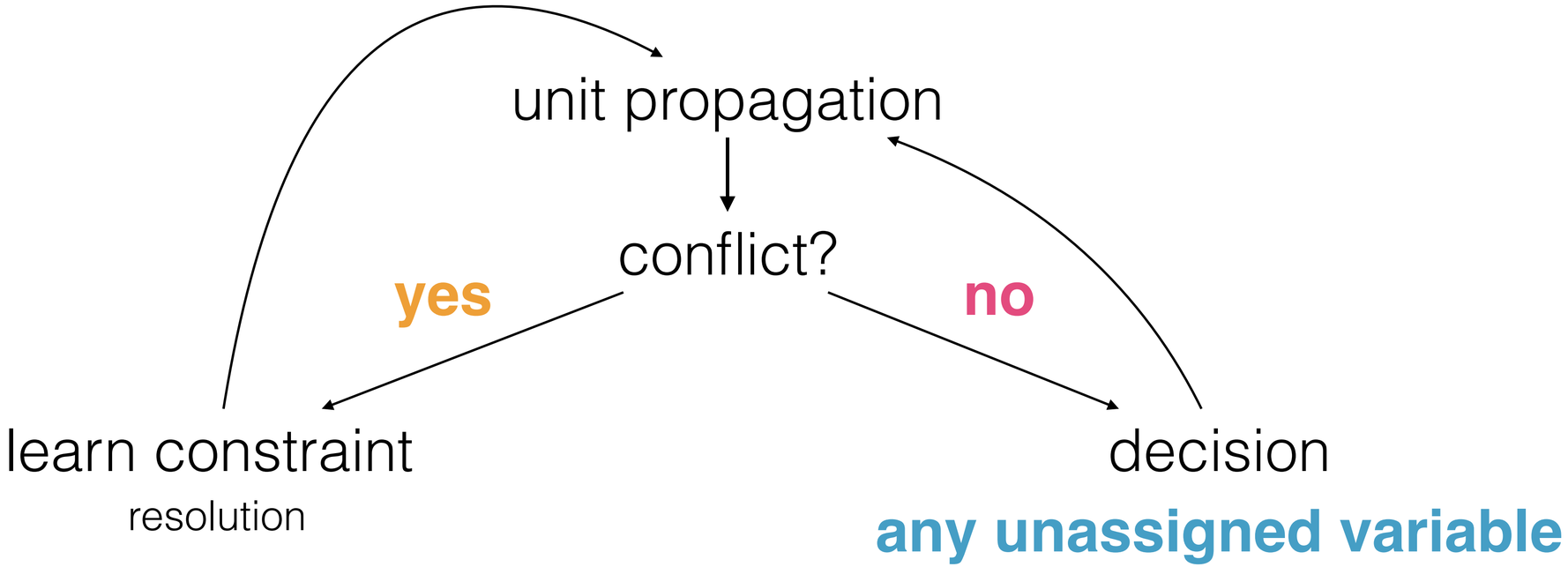
CDCL


unit propagation
conflict?
yes
no
learn constraint
resolution
decision

CDCL


unit propagation
conflict?
yes
no
learn constraint
resolution
decision

CDCL


unit propagation
conflict?
yes
no
learn constraint
resolution
decision
any unassigned variable

QCDCL

QCDCL

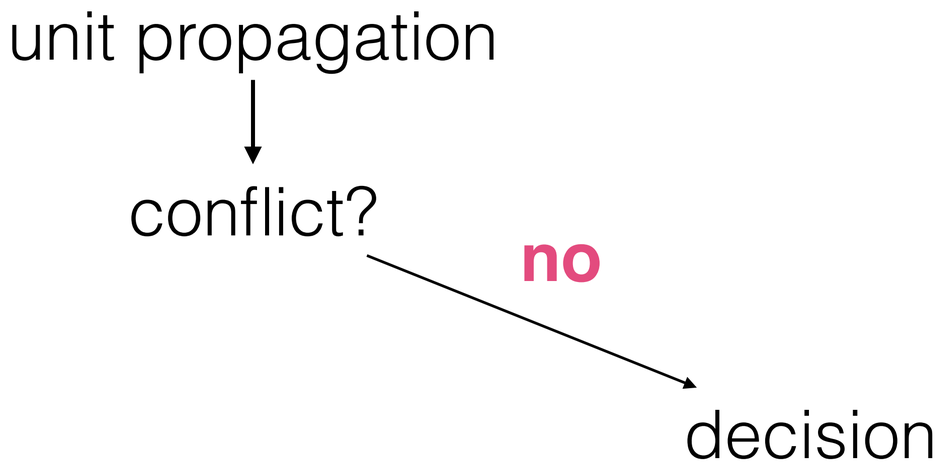
unit propagation

QCDCL

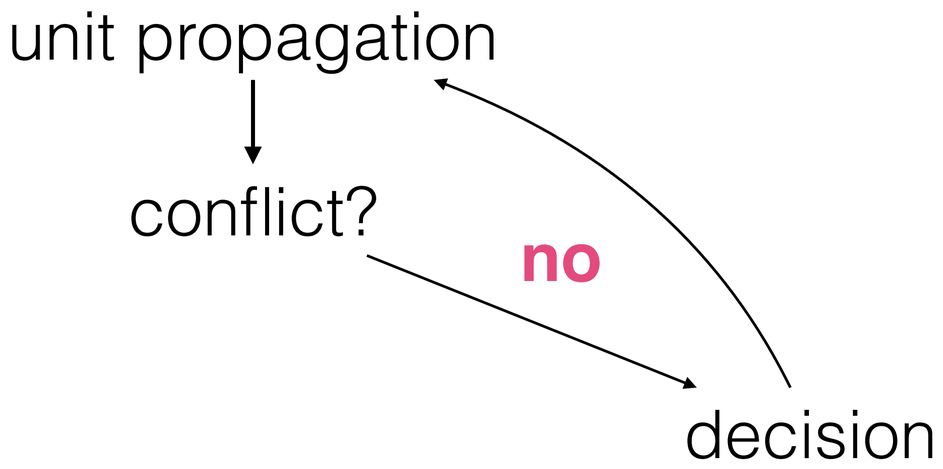
unit propagation

conflict?

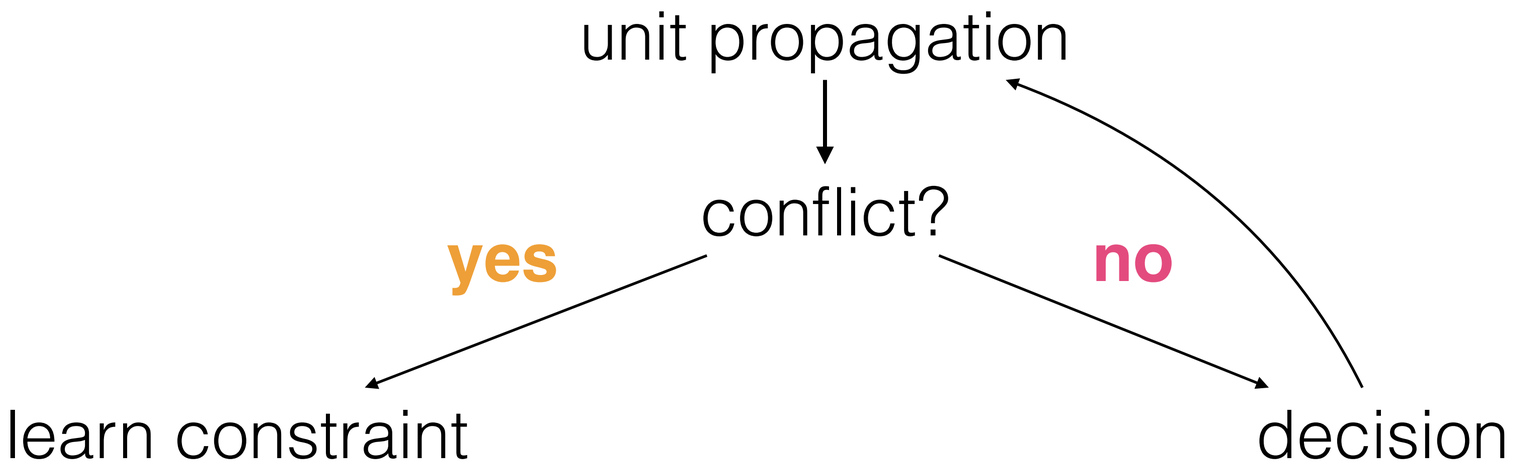
QCDCL



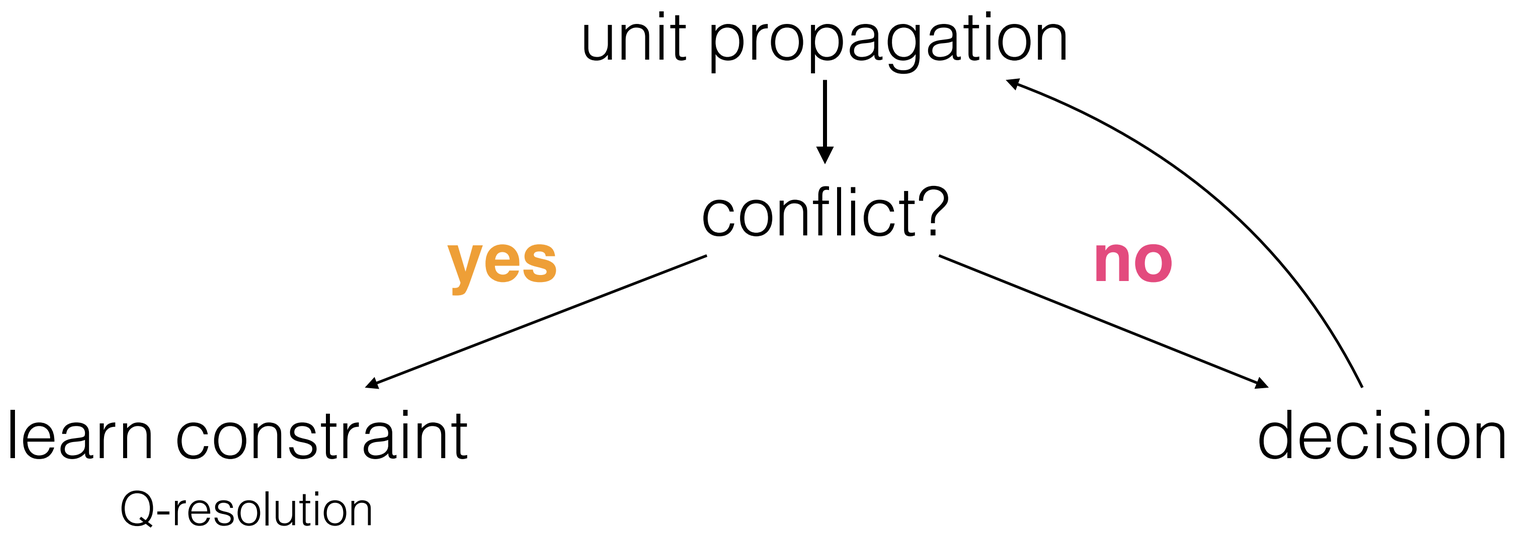
QCDCL



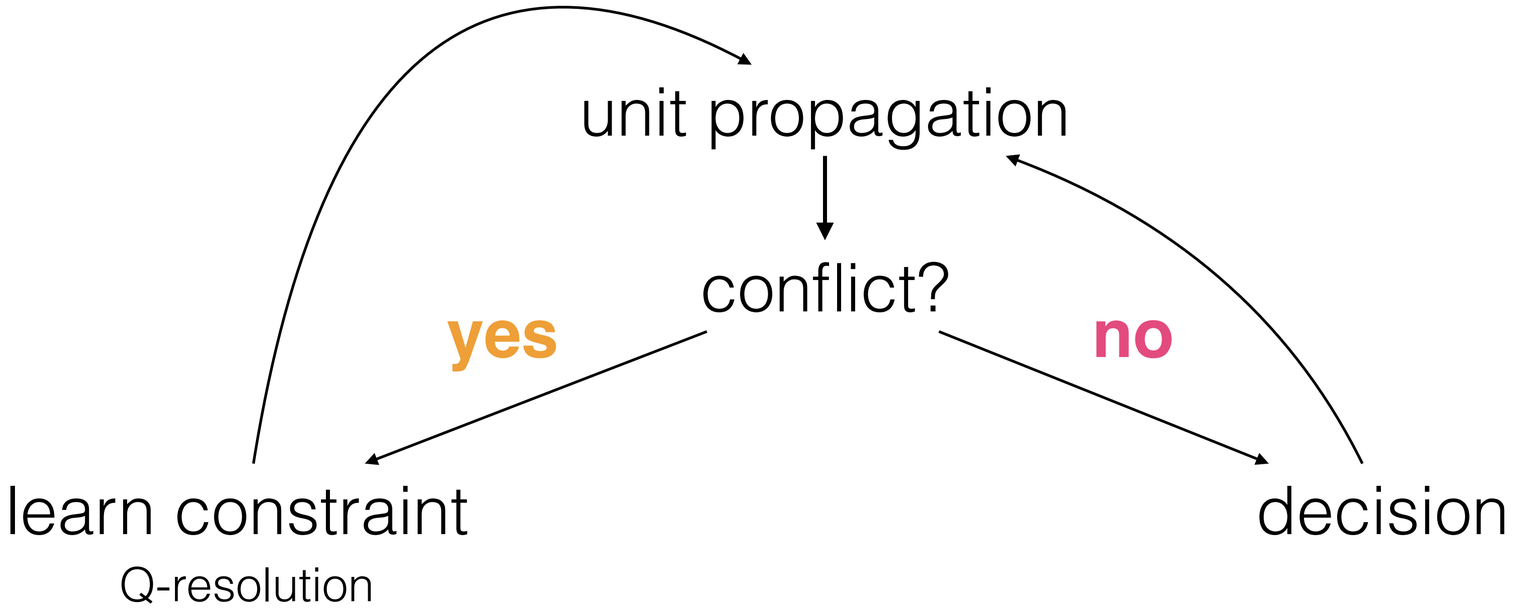
QCDCL


unit propagation
conflict?
yes
no
learn constraint
decision

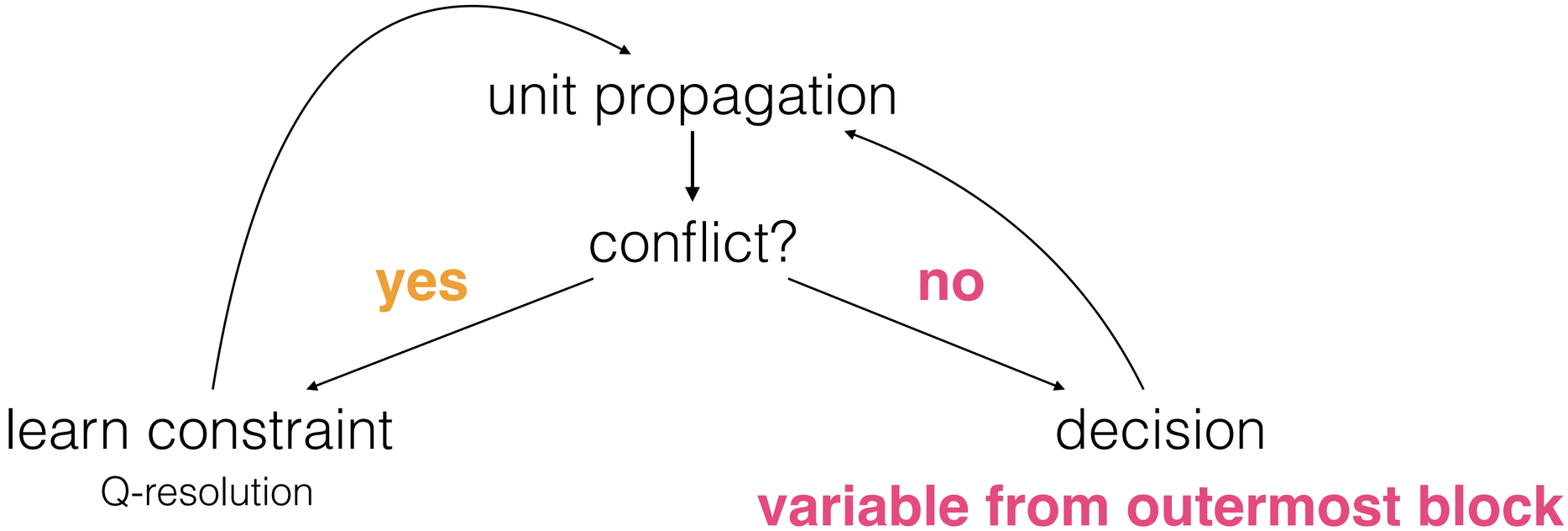
QCDCL


unit propagation
conflict?
yes
no
learn constraint
Q-resolution
decision

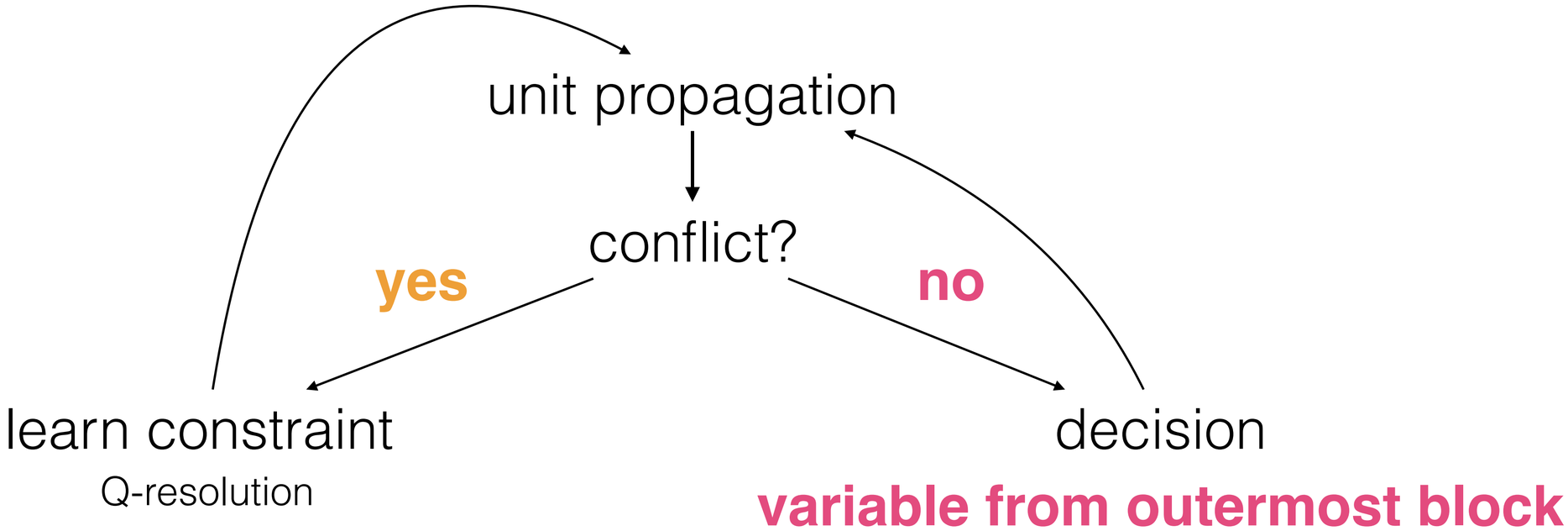
QCDCL


unit propagation
conflict?
yes
no
learn constraint
Q-resolution
decision

QCDCL

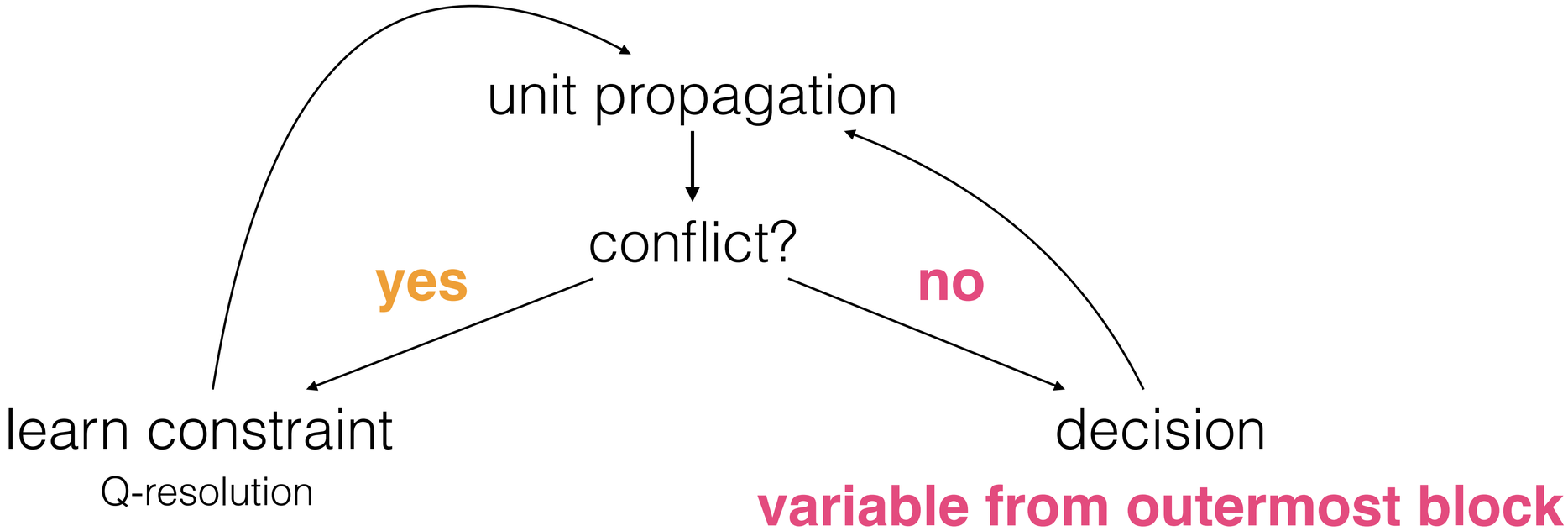

unit propagation
conflict?
yes
no
learn constraint
Q-resolution
decision
variable from outermost block

QCDCL



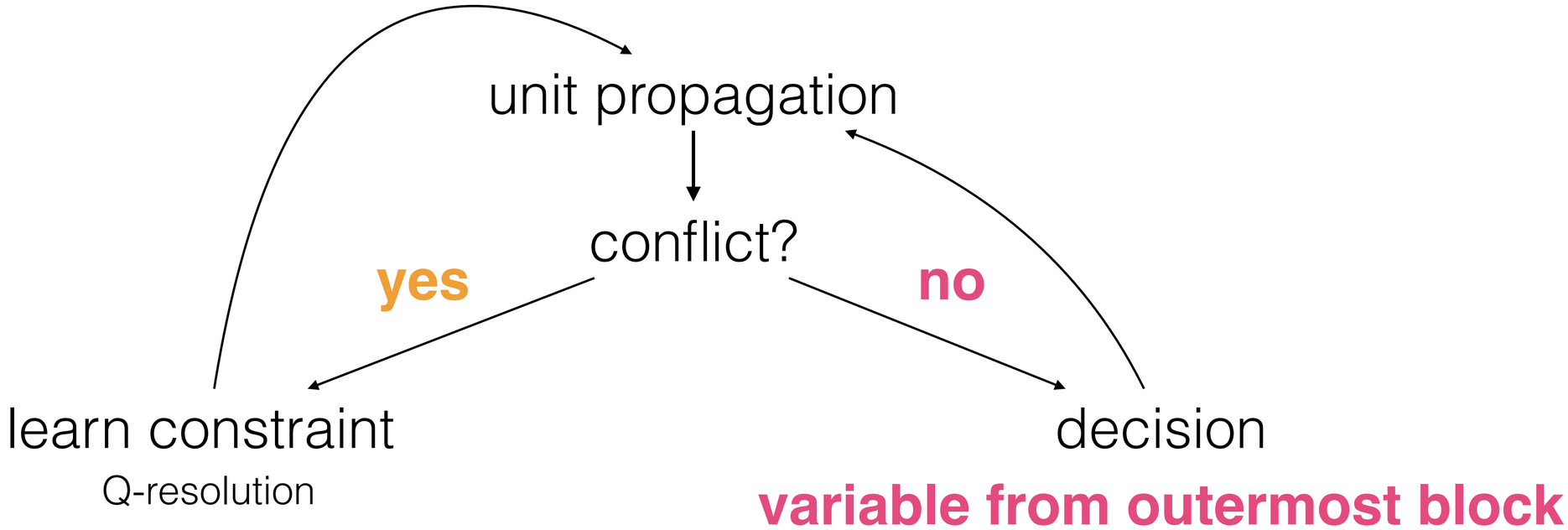
$$\forall X_1 \exists X_2 \forall X_3 \exists X_4 \ \ldots \ \forall X_{n-1} \exists X_n \ . \ \varphi$$

QCDCL



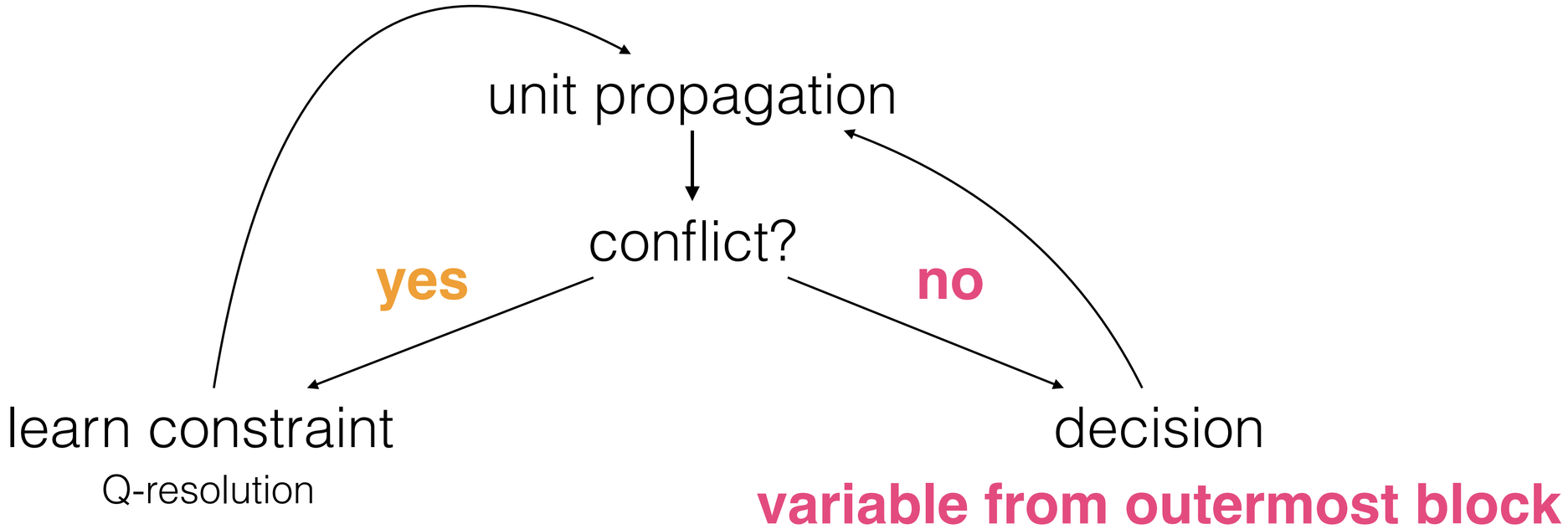
$\exists X_2 \forall X_3 \exists X_4 \ \ldots \ \forall X_{n-1} \exists X_n \ . \ \varphi$

QCDCL



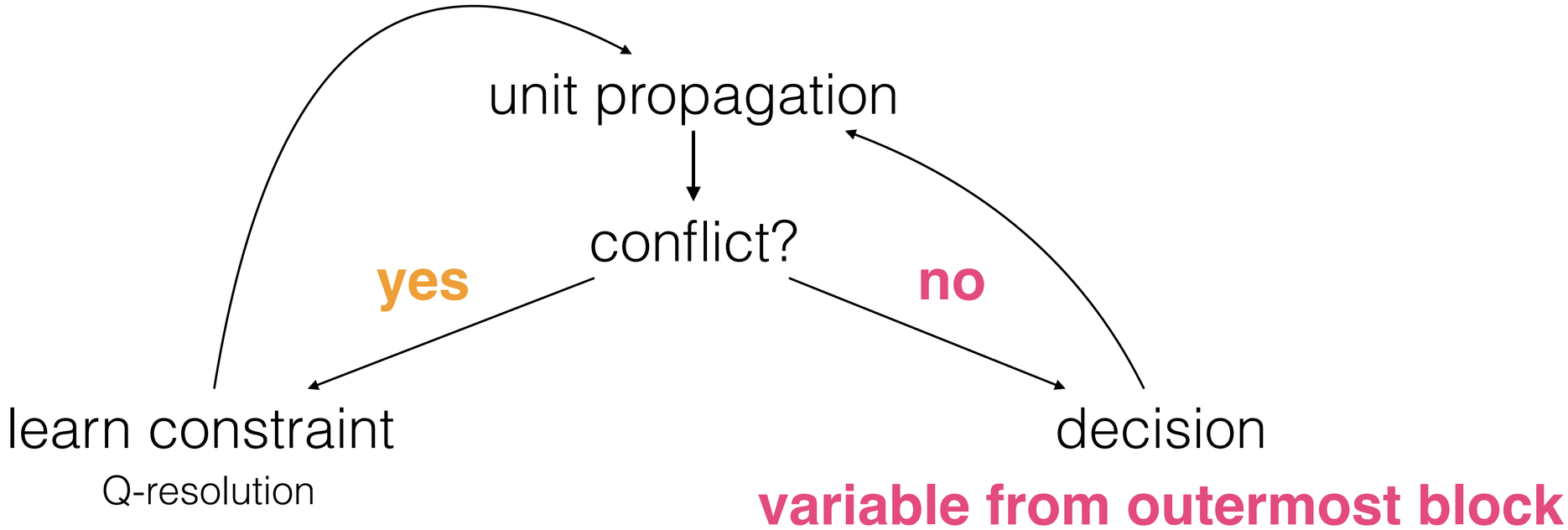
$\forall X_3 \exists X_4 \ \ldots \ \forall X_{n-1} \exists X_n \ . \ \varphi$

QCDCL



$\exists X_4 \ldots \forall X_{n-1} \exists X_n \,.\, \varphi$

QCDCL



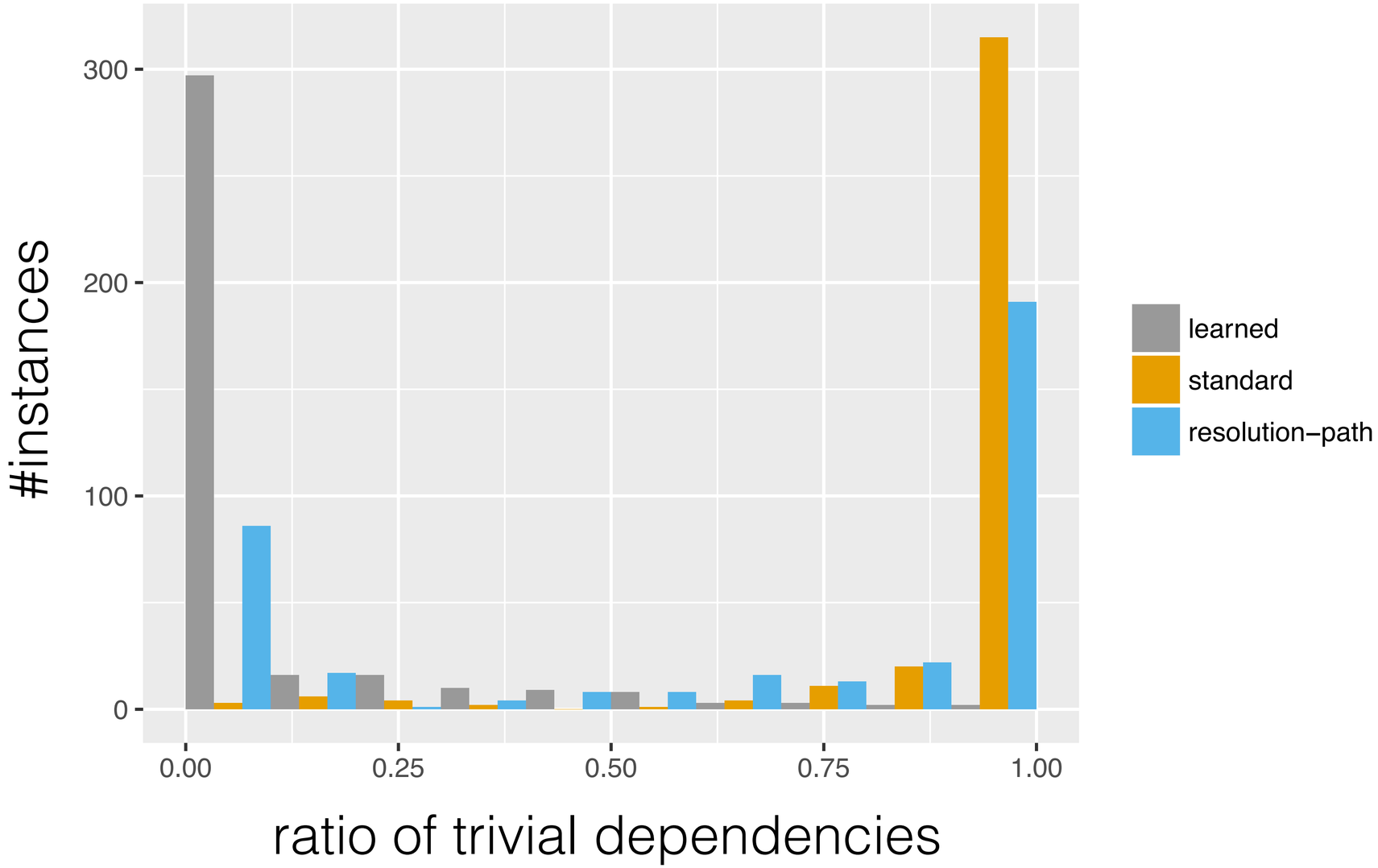
$\ldots \forall X_{n-1} \exists X_n \,.\, \varphi$

Peitl, S., Szeider: **Dependency Learning for QBF**. SAT 2017

Peitl, S., Szeider: **Dependency Learning for QBF**. SAT 2017

github.com/perebor/qute

Dependencies of preprocessed PCNF Instances

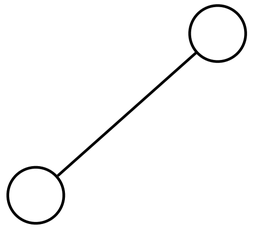


Use of (SAT) Oracles (QCDCL)

$Q_1x_1Q_2x_2Q_3x_3Q_4x_4 \ldots \varphi$ ○ **trail**

Use of (SAT) Oracles (QCDCL)

$Q_1x_1Q_2x_2Q_3x_3Q_4x_4 \ldots \varphi$

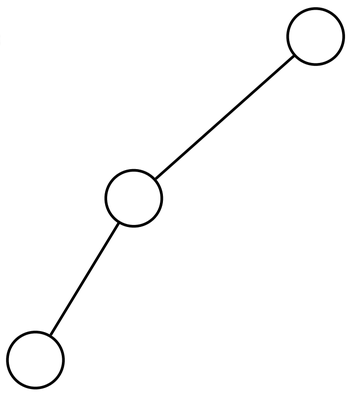


trail

x_1

Use of (SAT) Oracles (QCDCL)

$Q_1x_1Q_2x_2Q_3x_3Q_4x_4 \ldots \varphi$



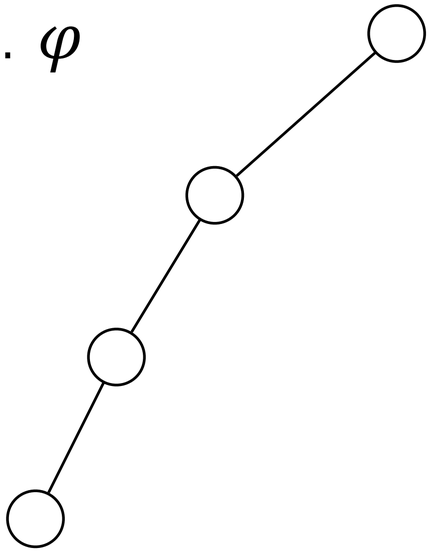
trail

x_1

$\neg x_2$

Use of (SAT) Oracles (QCDCL)

$Q_1x_1Q_2x_2Q_3x_3Q_4x_4 \ldots \varphi$



trail

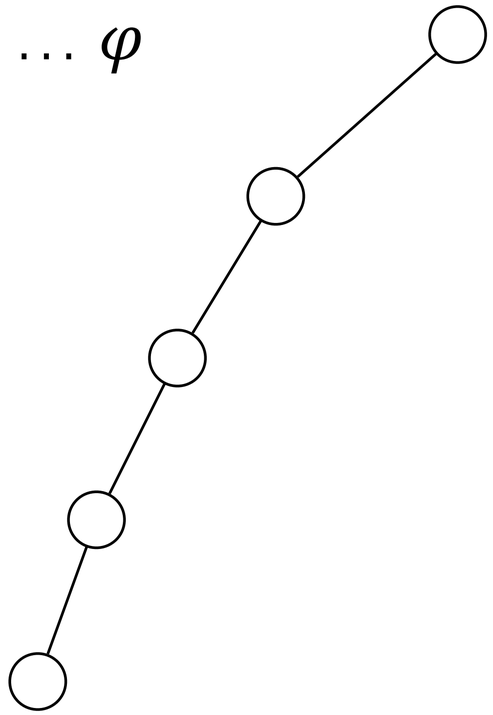
x_1

$\neg x_2$

$\neg x_3$

Use of (SAT) Oracles (QCDCL)

$Q_1x_1Q_2x_2Q_3x_3Q_4x_4 \ldots \varphi$



trail

x_1

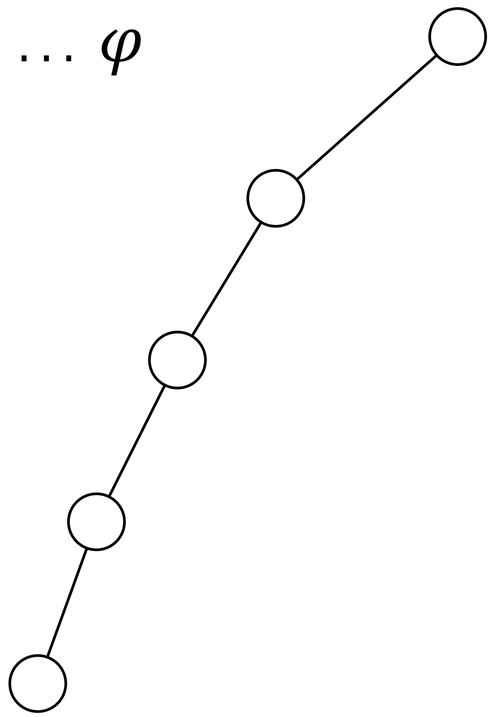
$\neg x_2$

$\neg x_3$

x_4

Use of (SAT) Oracles (QCDCL)

$Q_1x_1Q_2x_2Q_3x_3Q_4x_4 \ldots \varphi$



trail

x_1

$\neg x_2$

$\neg x_3$

x_4

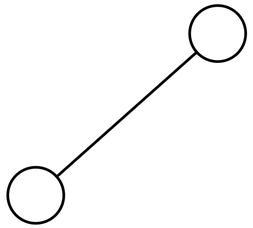
φ[**trail**] = **F** clause falsified

Use of (SAT) Oracles (QCDCL)

$Q_1x_1Q_2x_2Q_3x_3Q_4x_4 \ldots \varphi$ ○ **trail**

Use of (SAT) Oracles (QCDCL)

$Q_1x_1Q_2x_2Q_3x_3Q_4x_4 \ldots \varphi$

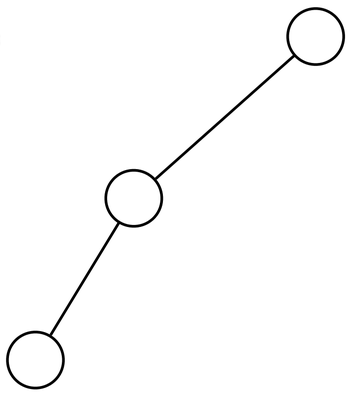


trail

x_1

Use of (SAT) Oracles (QCDCL)

$Q_1x_1Q_2x_2Q_3x_3Q_4x_4 \ldots \varphi$



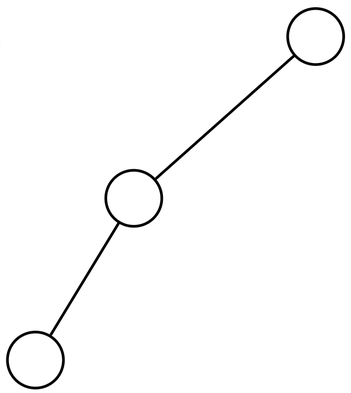
trail

x_1

$\neg x_2$

Use of (SAT) Oracles (QCDCL)

$Q_1x_1Q_2x_2Q_3x_3Q_4x_4 \ldots \varphi$



trail

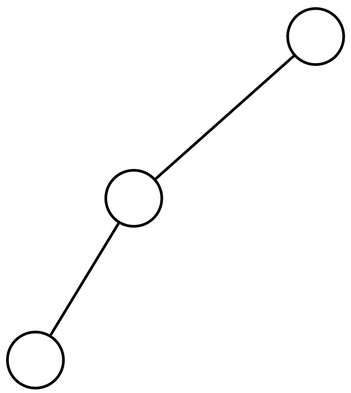
x_1

$\neg x_2$

$Q_3x_3Q_4x_4 \ldots \varphi[\textbf{trail}] = \textbf{F}$

Use of (SAT) Oracles (QCDCL)

$Q_1x_1Q_2x_2Q_3x_3Q_4x_4 \ldots \varphi$



trail

x_1

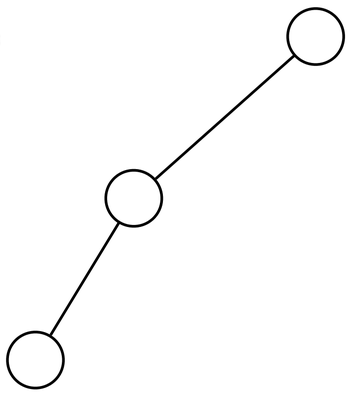
$\neg x_2$

$Q_3x_3Q_4x_4 \ldots \varphi[\textbf{trail}] = \textbf{F}$

- QBF Solver
- Preprocessor
- SAT Solver

Use of (SAT) Oracles (QCDCL)

$Q_1x_1Q_2x_2Q_3x_3Q_4x_4 \ldots \varphi$



trail

x_1

$\neg x_2$

$Q_3x_3Q_4x_4 \ldots \varphi[\textbf{trail}] = \textbf{F}$

- QBF Solver
- Preprocessor
- SAT Solver

Lonsing, Egly, Seidl: **Q-Resolution with Generalized Axioms**. SAT 2016

Use of (SAT) Oracles (CEGAR)

Use of (SAT) Oracles (CEGAR)

CEGAR2QBF($\forall X \exists Y . \varphi$):

Use of (SAT) Oracles (CEGAR)

```
CEGAR2QBF(∀X∃Y.φ):
  α = ∅
```

Use of (SAT) Oracles (CEGAR)

```
CEGAR2QBF(∀X∃Y.φ):
  α = ∅
  while True:
```

Use of (SAT) Oracles (CEGAR)

```
CEGAR2QBF(∀X∃Y.φ):
  α = ∅
  while True:
    τ = SAT(α)
```

Use of (SAT) Oracles (CEGAR)

```
CEGAR2QBF(∀X∃Y.φ):
  α = ∅
  while True:
    τ = SAT(α)
    if τ is None:
```

Use of (SAT) Oracles (CEGAR)

```
CEGAR2QBF(∀X∃Y.φ):
  α = ∅
  while True:
    τ = SAT(α)
    if τ is None:
      return True
```

Use of (SAT) Oracles (CEGAR)

```
CEGAR2QBF(∀X∃Y.φ):
  α = ∅
  while True:
    τ = SAT(α)
    if τ is None:
      return True
    else:
```

Use of (SAT) Oracles (CEGAR)

```
CEGAR2QBF(∀X∃Y.φ):
  α = ∅
  while True:
    τ = SAT(α)
    if τ is None:
      return True
    else:
      σ = SAT(φ(τ, Y))
```

Use of (SAT) Oracles (CEGAR)

```
CEGAR2QBF(∀X∃Y.φ):
  α = ∅
  while True:
    τ = SAT(α)
    if τ is None:
      return True
    else:
      σ = SAT(φ(τ, Y))
      if σ is None:
```

Use of (SAT) Oracles (CEGAR)

```
CEGAR2QBF(∀X∃Y.φ):
  α = ∅
  while True:
    τ = SAT(α)
    if τ is None:
      return True
    else:
      σ = SAT(φ(τ, Y))
      if σ is None:
        return False
```

Use of (SAT) Oracles (CEGAR)

```
CEGAR2QBF(∀X∃Y.φ):
  α = ∅
  while True:
    τ = SAT(α)
    if τ is None:
      return True
    else:
      σ = SAT(φ(τ, Y))
      if σ is None:
        return False
      else:
```

Use of (SAT) Oracles (CEGAR)

```
CEGAR2QBF(∀X∃Y.φ):
  α = ∅
  while True:
    τ = SAT(α)
    if τ is None:
      return True
    else:
      σ = SAT(φ(τ, Y))
      if σ is None:
        return False
      else:
        α = α ∧ φ(X, σ)
```

Use of (SAT) Oracles (CEGAR)

```
CEGAR2QBF(∀X∃Y.φ):
  α = ∅
  while True:
    τ = SAT(α)
    if τ is None:
      return True
    else:
      σ = SAT(φ(τ, Y))
      if σ is None:
        return False
      else:
        α = α ∧ φ(X, σ)
```

Janota et. al.: **Solving QBF with Counterexample Guided Refinement**. SAT 2012

Use of (SAT) Oracles (CEGAR)

$$\forall x_1 \ldots x_n \exists y_1 \ldots y_n \,.\, (x_1 = y_1) \wedge \ldots \wedge (x_n = y_n)$$

Use of (SAT) Oracles (CEGAR)

$$\forall x_1 \ldots x_n \exists y_1 \ldots y_n \,.\, (x_1 = y_1) \wedge \ldots \wedge (x_n = y_n)$$

$$\varnothing$$

Use of (SAT) Oracles (CEGAR)

$$\forall x_1 \ldots x_n \exists y_1 \ldots y_n \,.\, (x_1 = y_1) \wedge \ldots \wedge (x_n = y_n)$$

$\varnothing$ $\quad\quad\quad\quad$ T_1

Use of (SAT) Oracles (CEGAR)

$$\forall x_1 \ldots x_n \exists y_1 \ldots y_n \,.\, (x_1 = y_1) \wedge \ldots \wedge (x_n = y_n)$$

$\varnothing$ $\quad$ $\mathbf{\tau_1}$ $\quad$ $\mathbf{\sigma_1 = \tau_1}$

Use of (SAT) Oracles (CEGAR)

$$\forall x_1 \ldots x_n \exists y_1 \ldots y_n \,.\, (x_1 = y_1) \wedge \ldots \wedge (x_n = y_n)$$

$\varnothing$ $\qquad\qquad$ $\mathbf{\tau_1}$ $\qquad\qquad$ $\mathbf{\sigma_1} = \mathbf{\tau_1}$

$\wedge\ \varphi(X, \mathbf{\sigma_1})$

Use of (SAT) Oracles (CEGAR)

$$\forall x_1 \ldots x_n \exists y_1 \ldots y_n \,.\, (x_1 = y_1) \wedge \ldots \wedge (x_n = y_n)$$

$\varnothing$	$\mathbf{T_1}$	$\boldsymbol{\sigma_1} = \mathbf{T_1}$
$\wedge\ \varphi(X, \boldsymbol{\sigma_1})$	$\mathbf{T_2}$	

Use of (SAT) Oracles (CEGAR)

$$\forall x_1 \ldots x_n \exists y_1 \ldots y_n \,.\, (x_1 = y_1) \wedge \ldots \wedge (x_n = y_n)$$

$\varnothing$	$\mathbf{T_1}$	$\boldsymbol{\sigma_1} = \mathbf{T_1}$
$\wedge\ \varphi(X, \boldsymbol{\sigma_1})$	$\mathbf{T_2}$	$\boldsymbol{\sigma_2} = \mathbf{T_2}$

Use of (SAT) Oracles (CEGAR)

$$\forall x_1 \ldots x_n \exists y_1 \ldots y_n \,.\, (x_1 = y_1) \wedge \ldots \wedge (x_n = y_n)$$

$\varnothing$	$\mathbf{T_1}$	$\boldsymbol{\sigma_1} = \mathbf{T_1}$
$\wedge\ \varphi(X, \boldsymbol{\sigma_1})$	$\mathbf{T_2}$	$\boldsymbol{\sigma_2} = \mathbf{T_2}$
$\wedge\ \varphi(X, \boldsymbol{\sigma_2})$		

Use of (SAT) Oracles (CEGAR)

$$\forall x_1 \ldots x_n \exists y_1 \ldots y_n \,.\, (x_1 = y_1) \wedge \ldots \wedge (x_n = y_n)$$

$\varnothing$	$\mathbf{T_1}$	$\boldsymbol{\sigma_1} = \mathbf{T_1}$
$\wedge\ \varphi(X, \boldsymbol{\sigma_1})$	$\mathbf{T_2}$	$\boldsymbol{\sigma_2} = \mathbf{T_2}$
$\wedge\ \varphi(X, \boldsymbol{\sigma_2})$	$\mathbf{T_3}$	

Use of (SAT) Oracles (CEGAR)

$$\forall x_1 \ldots x_n \exists y_1 \ldots y_n \,.\, (x_1 = y_1) \wedge \ldots \wedge (x_n = y_n)$$

$\varnothing$	$\mathbf{T_1}$	$\boldsymbol{\sigma_1} = \mathbf{T_1}$
$\wedge\ \varphi(X, \boldsymbol{\sigma_1})$	$\mathbf{T_2}$	$\boldsymbol{\sigma_2} = \mathbf{T_2}$
$\wedge\ \varphi(X, \boldsymbol{\sigma_2})$	$\mathbf{T_3}$	$\boldsymbol{\sigma_3} = \mathbf{T_3}$

Use of (SAT) Oracles (CEGAR)

$$\forall x_1 \ldots x_n \exists y_1 \ldots y_n \,.\, (x_1 = y_1) \wedge \ldots \wedge (x_n = y_n)$$

$\varnothing$	$\mathbf{T_1}$	$\boldsymbol{\sigma_1} = \mathbf{T_1}$
$\wedge\ \varphi(X, \boldsymbol{\sigma_1})$	$\mathbf{T_2}$	$\boldsymbol{\sigma_2} = \mathbf{T_2}$
$\wedge\ \varphi(X, \boldsymbol{\sigma_2})$	$\mathbf{T_3}$	$\boldsymbol{\sigma_3} = \mathbf{T_3}$
	…	

Lower Bounds by Strategy Extraction

Lower Bounds by Strategy Extraction

Beyersdorff, Blinkhorn: **Formulas with Large Weight: a New Technique for Genuine QBF Lower Bounds**. STACS 2018

Lower Bounds by Strategy Extraction

Beyersdorff, Blinkhorn: **Formulas with Large Weight: a New Technique for Genuine QBF Lower Bounds**. STACS 2018

Minimum **range** of a strategy $2^X \to 2^Y$ is a lower bound on proof size.

Lower Bounds by Strategy Extraction

Beyersdorff, Blinkhorn: **Formulas with Large Weight: a New Technique for Genuine QBF Lower Bounds**. STACS 2018

Minimum **range** of a strategy $2^X \to 2^Y$ is a lower bound on proof size.

Beyersdorff, Chew, Janota: **Proof Complexity of Resolution-based QBF Calculi**. STACS 2015

Lower Bounds by Strategy Extraction

Beyersdorff, Blinkhorn: **Formulas with Large Weight: a New Technique for Genuine QBF Lower Bounds**. STACS 2018

Minimum **range** of a strategy $2^X \to 2^Y$ is a lower bound on proof size.

Beyersdorff, Chew, Janota: **Proof Complexity of Resolution-based QBF Calculi**. STACS 2015

Strategies extracted from QCDCL proofs are **bounded depth circuits**.

Machine Learning for Generalization

$$\forall X \exists Y . \varphi$$

Machine Learning for Generalization

$$\forall X \exists Y . \varphi$$

T_1 σ_1

Machine Learning for Generalization

$$\forall X \exists Y . \varphi$$

T_1 σ_1

T_2 σ_2

Machine Learning for Generalization

$$\forall X \exists Y . \varphi$$

T_1 σ_1

T_2 σ_2

T_3 σ_3

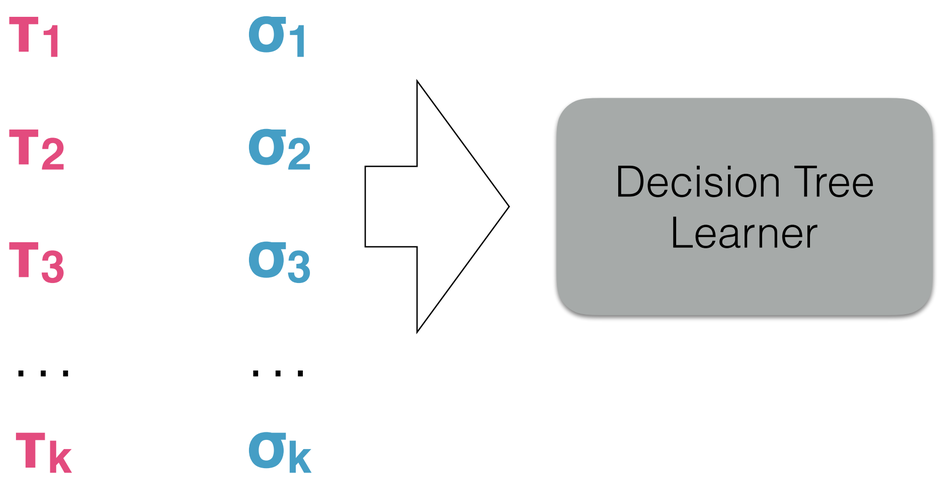
Machine Learning for Generalization

$$\forall X \exists Y . \varphi$$

T_1	σ_1
T_2	σ_2
T_3	σ_3
…	…
T_k	σ_k

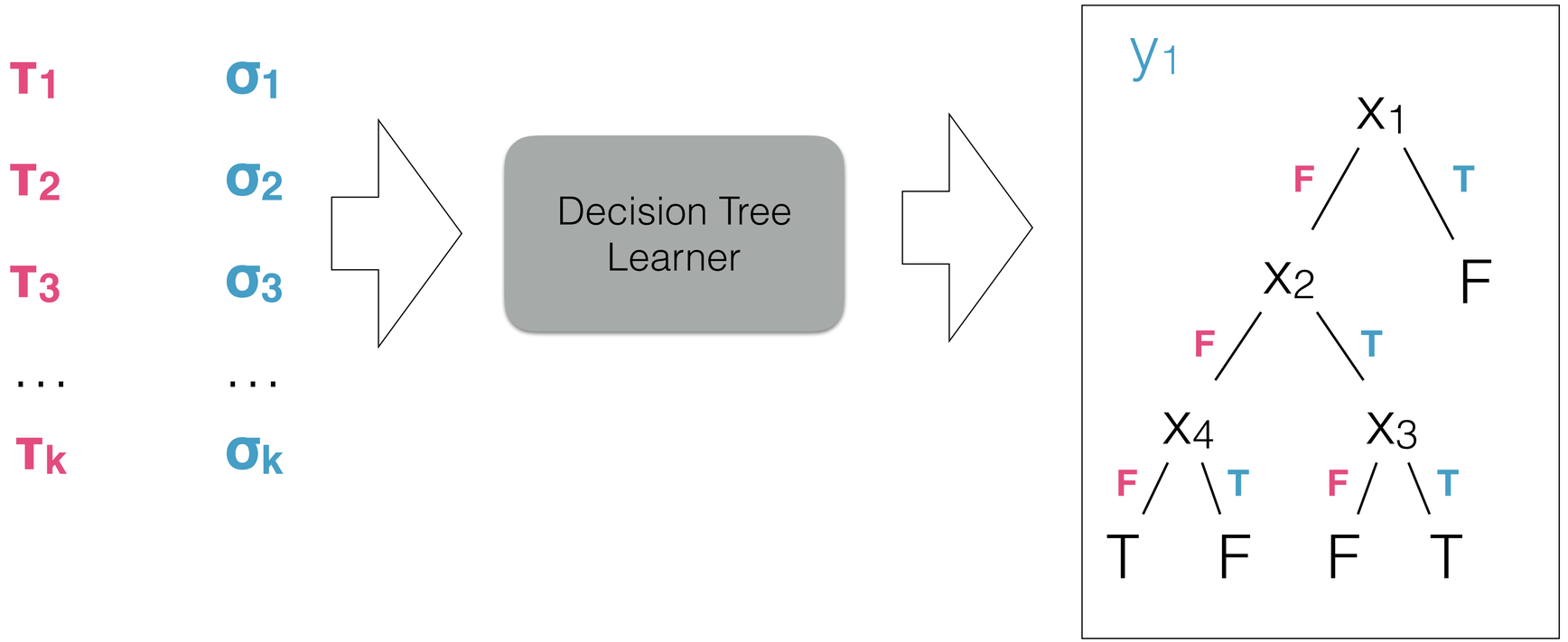
Machine Learning for Generalization

$\forall X \exists Y . \varphi$



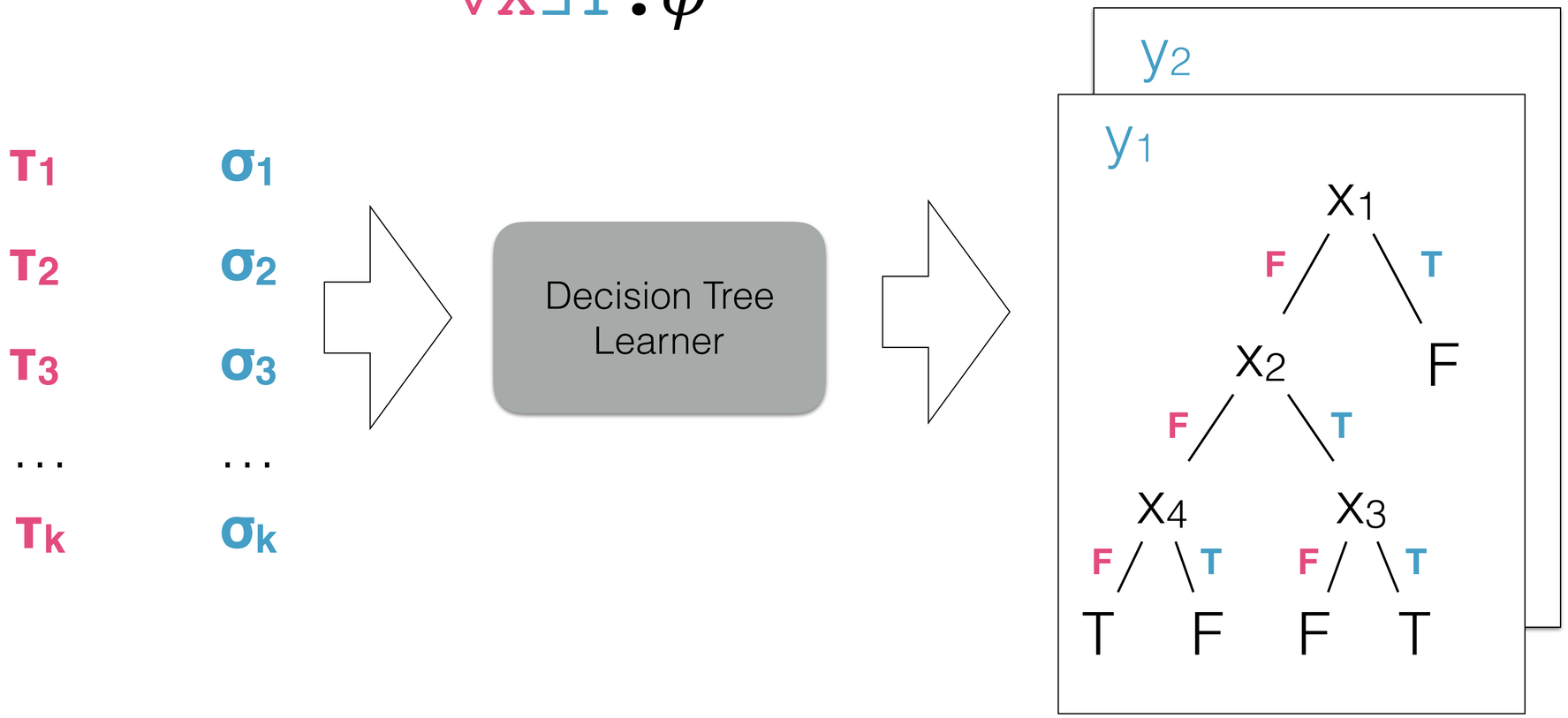
Machine Learning for Generalization

$\forall X \exists Y . \varphi$



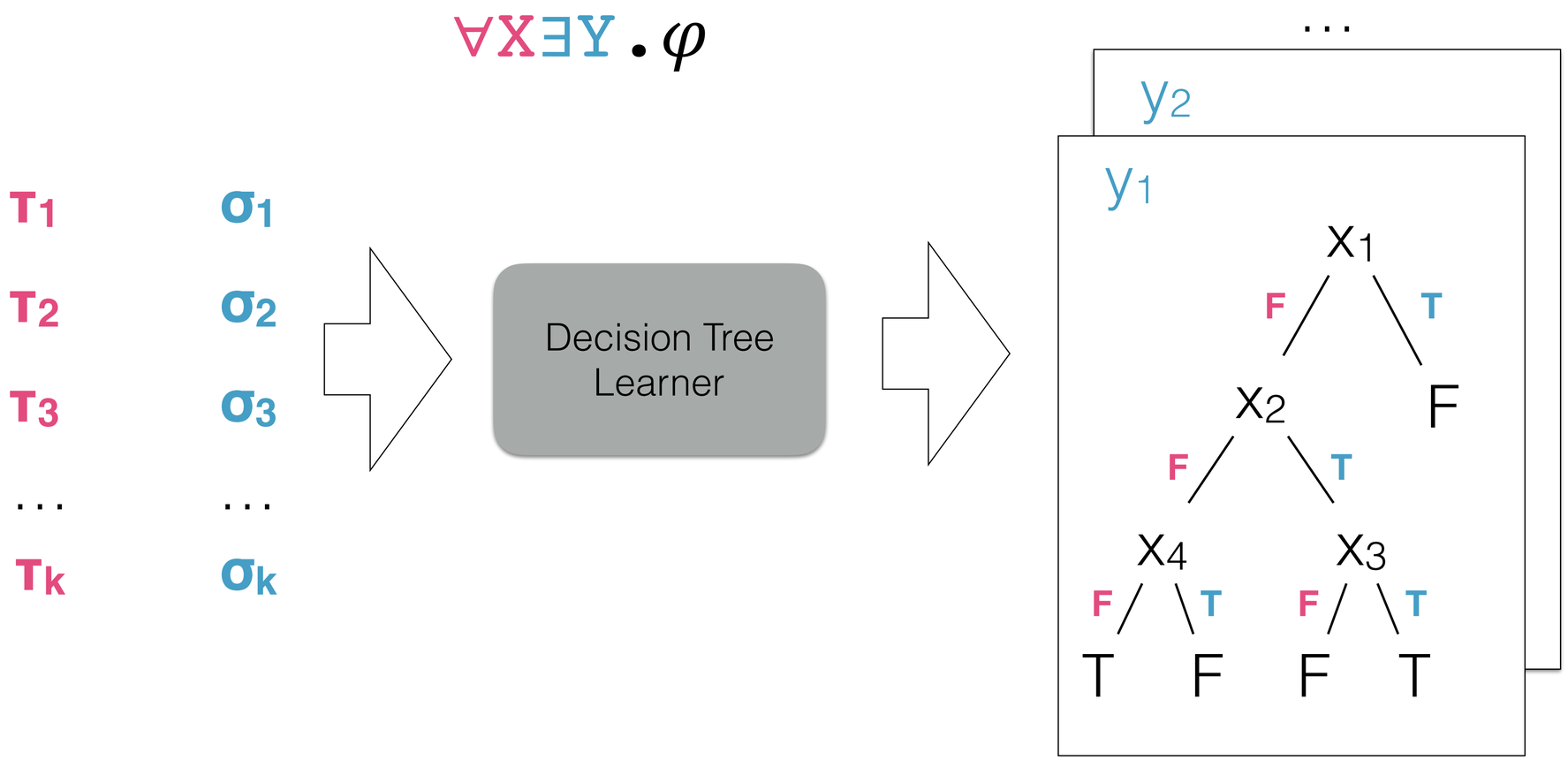
Machine Learning for Generalization

$\forall X \exists Y . \varphi$

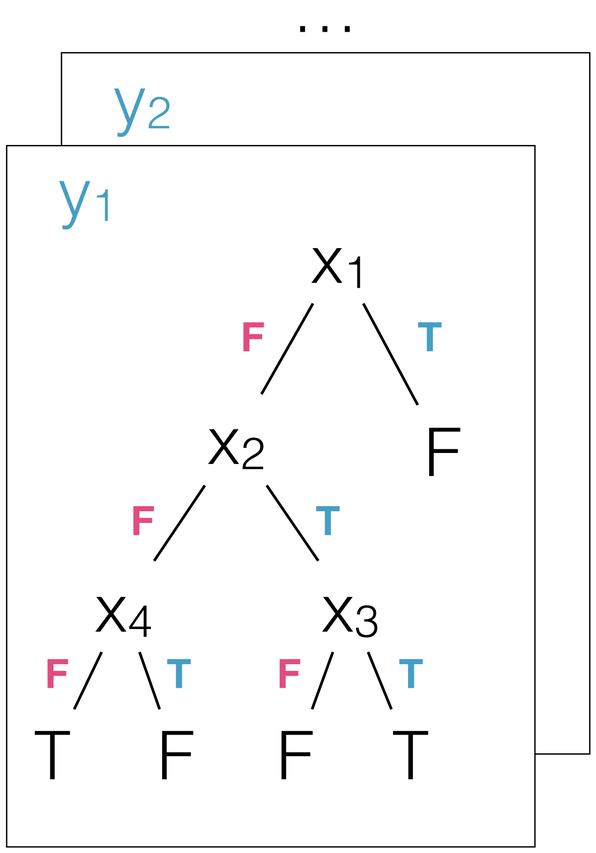


Machine Learning for Generalization

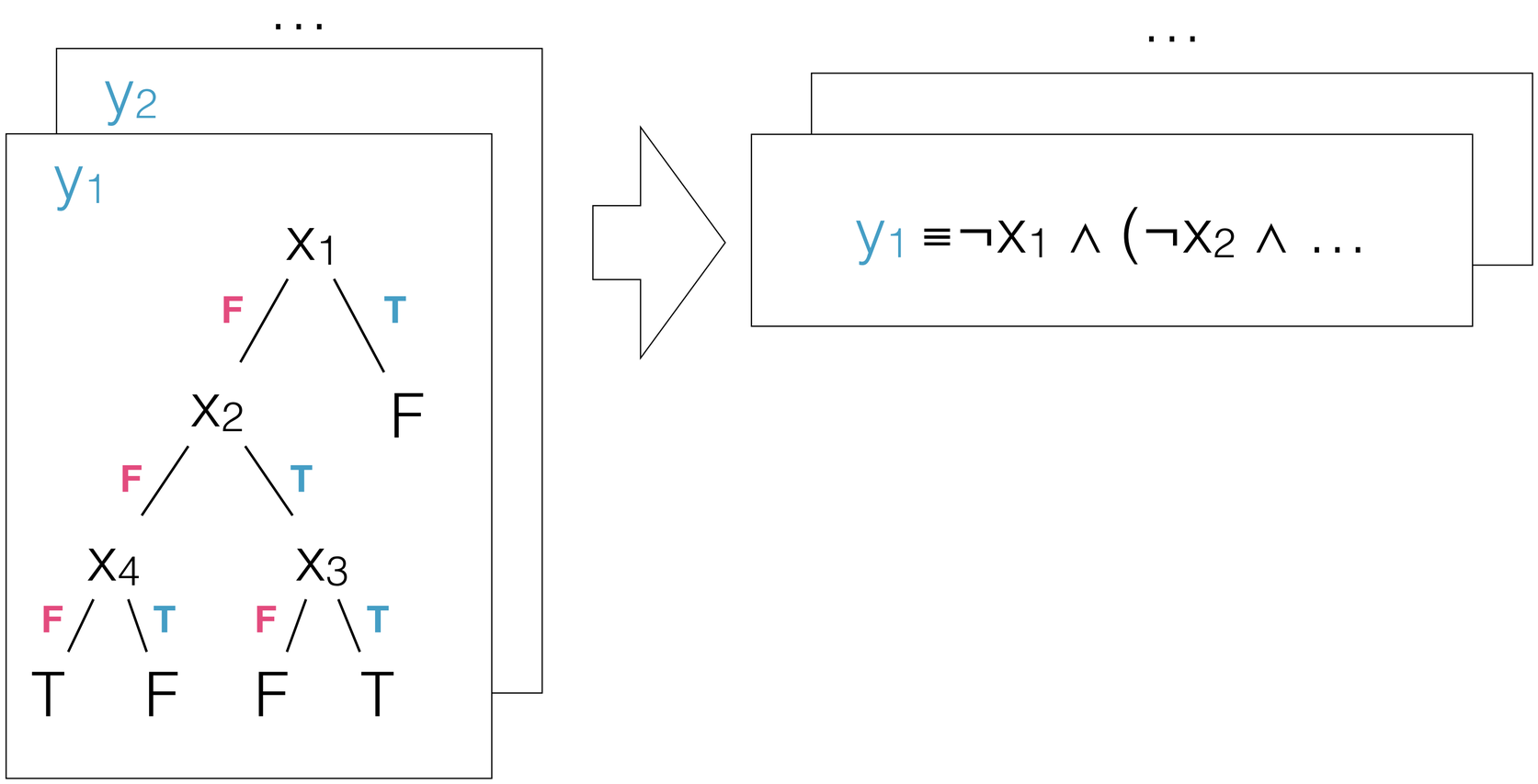
$\forall X \exists Y . \varphi$



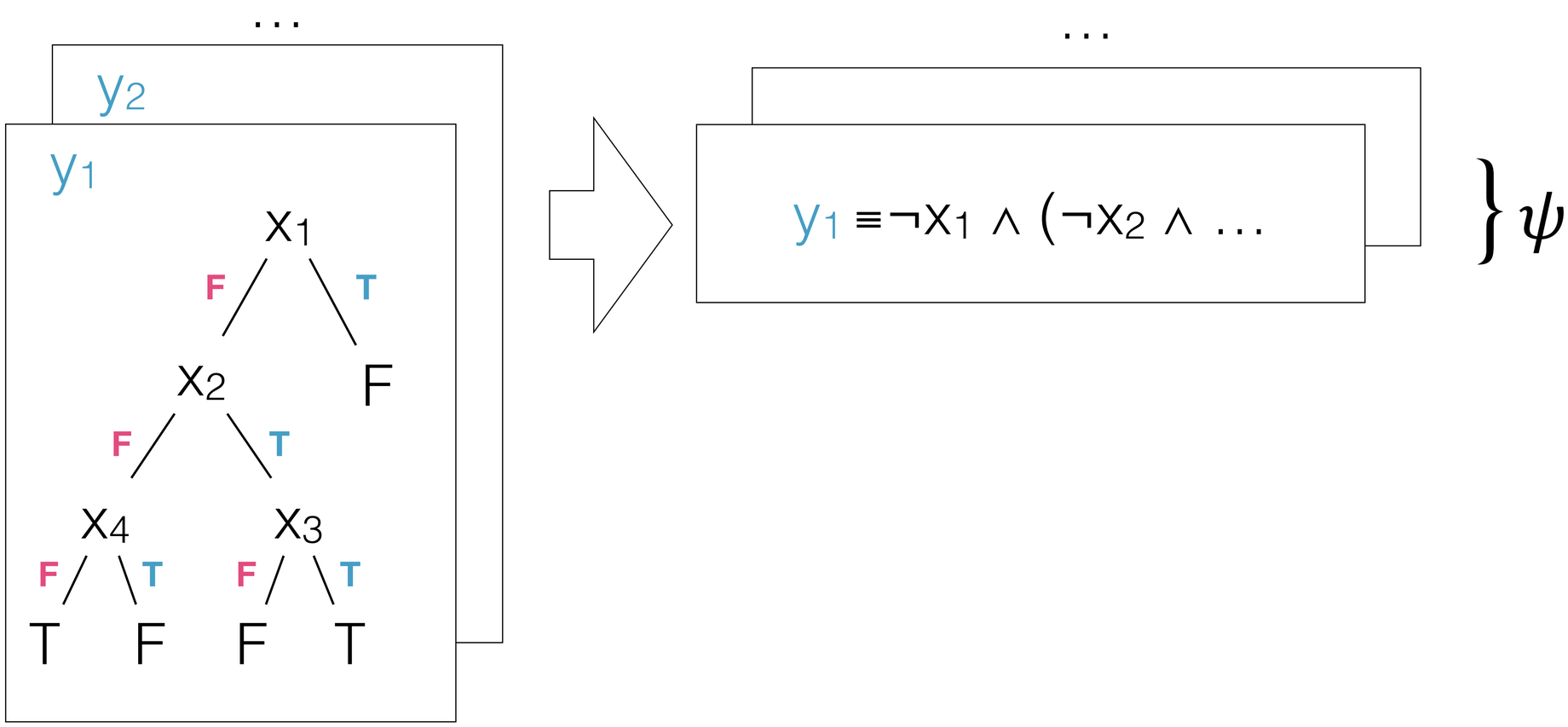
Machine Learning for Generalization



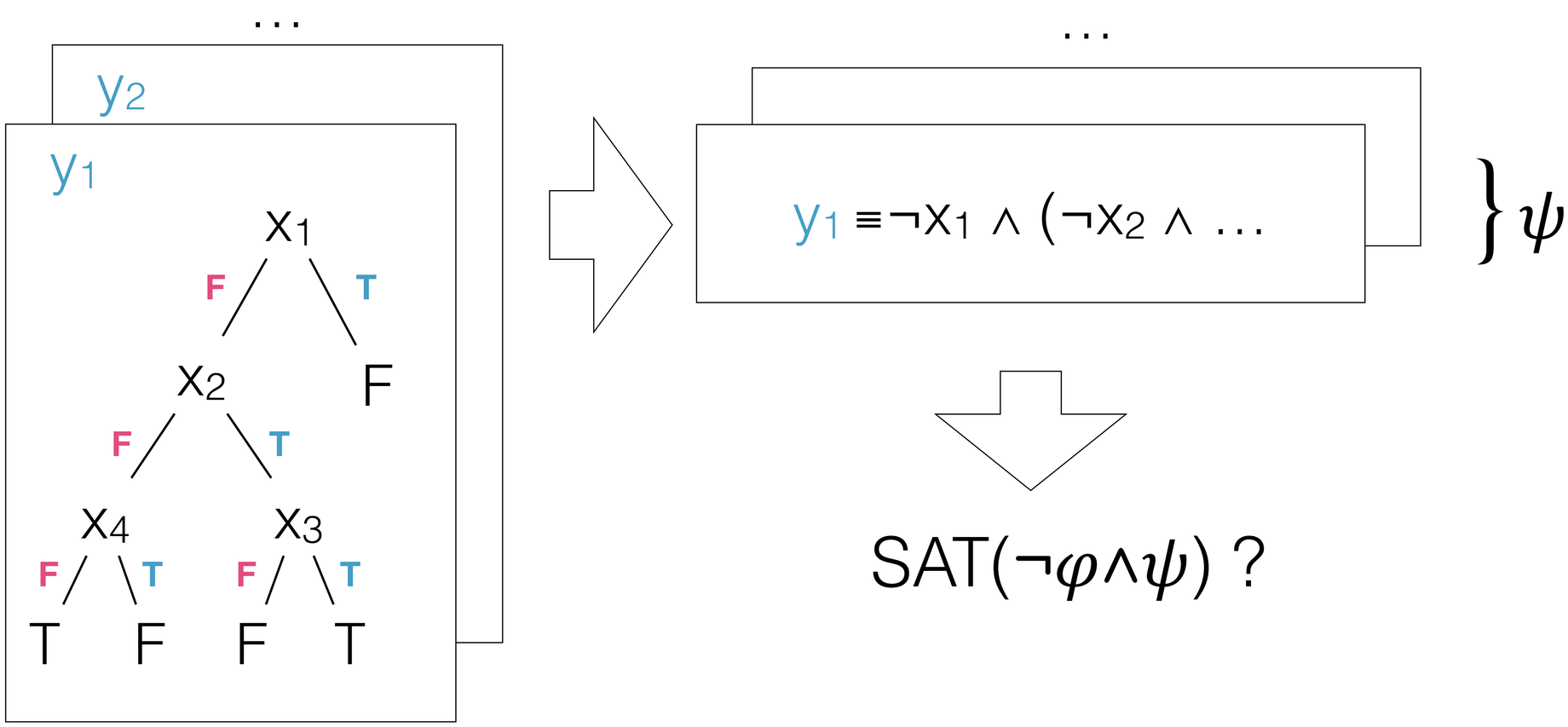
Machine Learning for Generalization



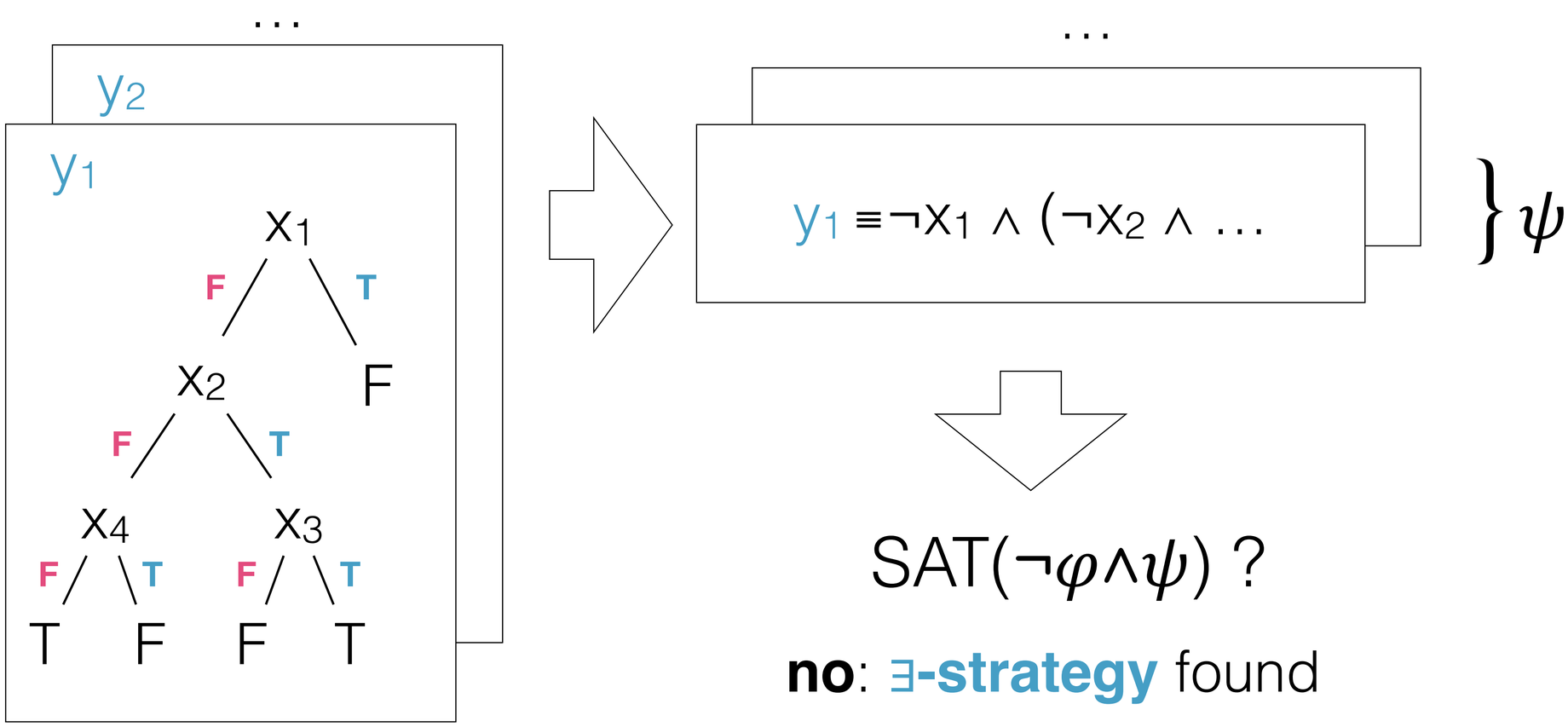
Machine Learning for Generalization



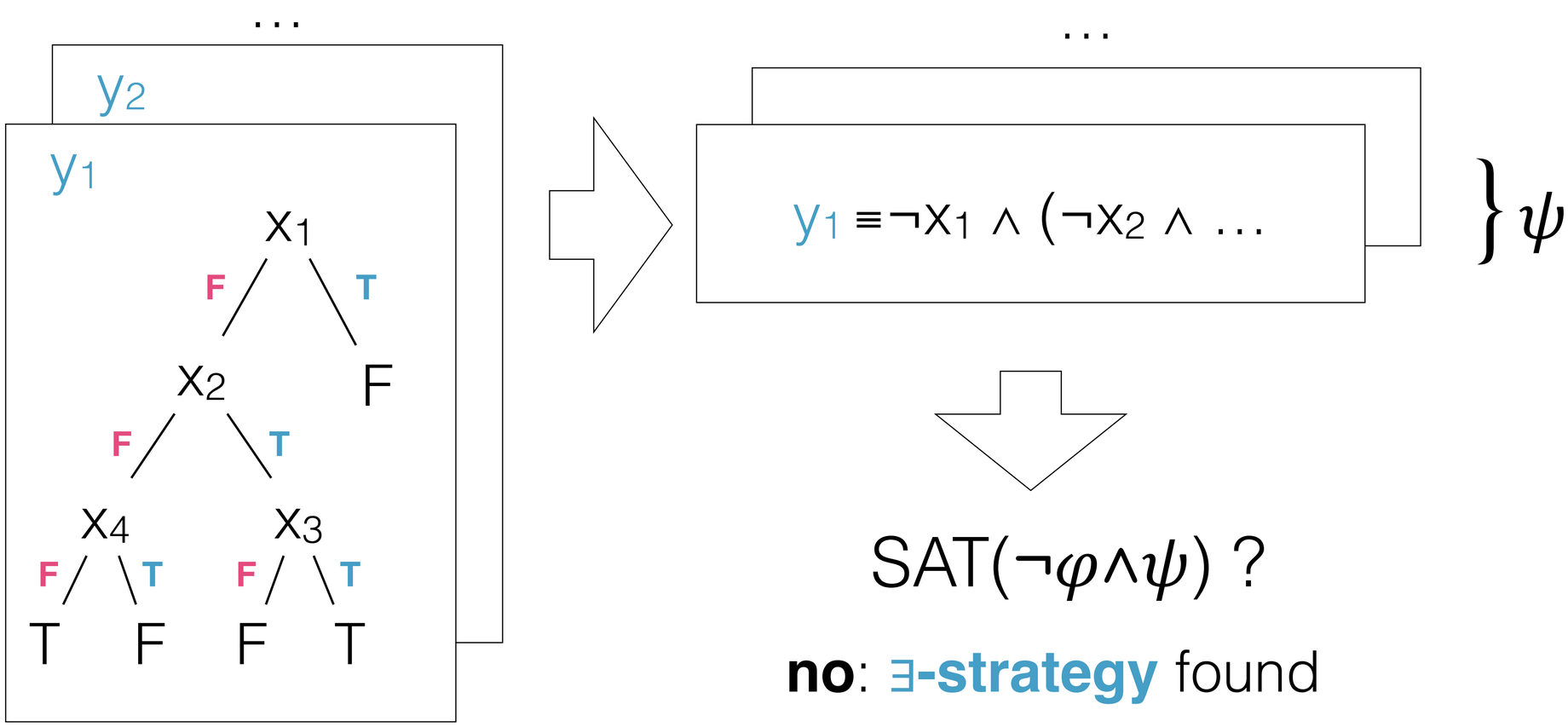
Machine Learning for Generalization



Machine Learning for Generalization



Machine Learning for Generalization



Janota: **Towards Generalization in QBF Solving via Machine Learning**. AAAI 2018

Incremental Determinization

$\forall X \exists Y . \varphi$

Incremental Determinization

$$\forall X \exists Y . \varphi$$

$$(\mathbf{x} \vee \neg \mathbf{y_1}) \wedge (\neg \mathbf{x} \vee \mathbf{y_1})$$

Incremental Determinization

$$\forall X \exists Y . \varphi$$

$(\mathbf{x} \vee \neg \mathbf{y_1}) \wedge (\neg \mathbf{x} \vee \mathbf{y_1})$ definition of $\mathbf{y_1}$ in terms of X

Incremental Determinization

$\forall X \exists Y . \varphi$

$(\mathbf{x} \vee \neg \mathbf{y_1}) \wedge (\neg \mathbf{x} \vee \mathbf{y_1})$ definition of $\mathbf{y_1}$ in terms of X

$(\mathbf{x} \vee \neg \mathbf{y_2}) \wedge (\neg \mathbf{y_1} \vee \mathbf{y_2})$

Incremental Determinization

$$\forall X \exists Y . \varphi$$

$(\mathbf{x} \vee \neg \mathbf{y_1}) \wedge (\neg \mathbf{x} \vee \mathbf{y_1})$ definition of $\mathbf{y_1}$ in terms of X

$(\mathbf{x} \vee \neg \mathbf{y_2}) \wedge (\neg \mathbf{y_1} \vee \mathbf{y_2})$ definition of $\mathbf{y_2}$ in terms of X and $\mathbf{y_1}$

Incremental Determinization

$$\forall X \exists Y . \varphi$$

$(\mathbf{x} \vee \neg \mathbf{y_1}) \wedge (\neg \mathbf{x} \vee \mathbf{y_1})$ definition of $\mathbf{y_1}$ in terms of X

$(\mathbf{x} \vee \neg \mathbf{y_2}) \wedge (\neg \mathbf{y_1} \vee \mathbf{y_2})$ definition of $\mathbf{y_2}$ in terms of X and $\mathbf{y_1}$

- maintain growing set of definitions

Incremental Determinization

$$\forall X \exists Y . \varphi$$

$(\mathbf{x} \vee \neg \mathbf{y_1}) \wedge (\neg \mathbf{x} \vee \mathbf{y_1})$ definition of $\mathbf{y_1}$ in terms of X

$(\mathbf{x} \vee \neg \mathbf{y_2}) \wedge (\neg \mathbf{y_1} \vee \mathbf{y_2})$ definition of $\mathbf{y_2}$ in terms of X and $\mathbf{y_1}$

- maintain growing set of definitions
- try adding clauses to make **y**'s determined (branching)

Incremental Determinization

$$\forall X \exists Y . \varphi$$

$(\mathbf{x} \vee \neg \mathbf{y_1}) \wedge (\neg \mathbf{x} \vee \mathbf{y_1})$ definition of $\mathbf{y_1}$ in terms of X

$(\mathbf{x} \vee \neg \mathbf{y_2}) \wedge (\neg \mathbf{y_1} \vee \mathbf{y_2})$ definition of $\mathbf{y_2}$ in terms of X and $\mathbf{y_1}$

- maintain growing set of definitions
- try adding clauses to make **y**'s determined (branching)

Rabe, Seshia: **Incremental Determinization**. SAT 2016

Wrap-Up

Wrap-Up

QBF solvers are becoming **more effective**

Wrap-Up

QBF solvers are becoming **more effective**

room (and a need) for **new ideas**

Wrap-Up

QBF solvers are becoming **more effective**

room (and a need) for **new ideas**

success hinges on **applications** (in verification/synthesis)

Wrap-Up

QBF solvers are becoming **more effective**

room (and a need) for **new ideas**

success hinges on **applications** (in verification/synthesis)

Workshop on QBFs and Beyond @ **FloC 2018**