Exploiting Partial Duality in QBF

Alexandra Goultiaeva and Fahiem Bacchus

University of Toronto, Canada

SAT 2013

Outline

- Background
  - Dual propagation
  - Dual propagation in existing CNF solvers
- Partial duality
  - Using reconstructed gates
  - Reconstructing Plaisted-Greenbaum
- Experiments and conclusion

CNF representation

- Handles conflicts well, but loses information about solutions

CNF representation

- Handles conflicts well, but loses information about solutions
- Example:

$$\exists e \forall x_1 x_2 x_3 \ldots x_n$$

$$\left(\left(x_1 \neq x_2\right) \vee \left(x_2 \neq x_3\right) \vee \left(x_3 \neq x_4\right) \vee \ldots\right)$$

$$\vee f\left(x_{1,} x_{2,} x_{3,} x_{4,} e\right)$$

CNF representation

- Handles conflicts well, but loses information about solutions
- Example:

$$\exists e \forall x_1 x_2 x_3 \ldots x_n$$

$$\left(\left(x_1 \neq x_2\right) \vee \left(x_2 \neq x_3\right) \vee \left(x_3 \neq x_4\right) \vee \ldots\right)$$

$$\vee f\left(x_{1,} x_{2,} x_{3,} x_{4,} e\right)$$

Tseitin

$$\exists e \forall x_1 x_2 x_3 \ldots x_n \ \exists g_1 g_2 g_3 \ldots$$

$$\left(g_1 \vee g_2 \vee g_3 \vee \ldots\right)$$

$$g_1 \equiv \left(x_1 \neq x_2\right)$$

$$g_2 \equiv \left(x_2 \neq x_3\right)$$

$$\ldots$$

$$g_k \equiv f\left(x_{1,} \ldots, e\right)$$

CNF representation

- Handles conflicts well, but loses information about solutions

$$\exists e \forall x_1 x_2 x_3 \ldots x_n$$

$$\left(\left(x_1 \neq x_2\right) \vee \left(x_2 \neq x_3\right) \vee \left(x_3 \neq x_4\right) \vee \ldots\right)$$

$$\vee f\left(x_{1,} x_{2,} x_{3,} x_{4,} e\right)$$

Possible solution:

$$\left(x_1 \wedge \neg x_2\right)$$

Tseitin

$$\exists e \forall x_1 x_2 x_3 \ldots x_n \; \exists g_1 g_2 g_3 \ldots$$

$$\left(g_1 \vee g_2 \vee g_3 \vee \ldots\right)$$

$$g_1 \equiv \left(x_1 \neq x_2\right)$$

$$g_2 \equiv \left(x_2 \neq x_3\right)$$

$$\ldots$$

$$g_k \equiv f\left(x_{1,} \ldots, e\right)$$

CNF representation

- Handles conflicts well, but loses information about solutions

$$\exists e \forall x_1 x_2 x_3 \ldots x_n$$
$$\left(\left(x_1 \neq x_2\right) \vee \left(x_2 \neq x_3\right) \vee \left(x_3 \neq x_4\right) \vee \ldots\right)$$
$$\vee f\left(x_{1,} x_{2,} x_{3,} x_{4,} e\right)$$

Possible solution:

$$\left(x_1 \wedge \neg x_2\right)$$

Not verifiable in Tseitin

Tseitin

$$\exists e \forall x_1 x_2 x_3 \ldots x_n \; \exists g_1 g_2 g_3 \ldots$$

$$\left(g_1 \vee g_2 \vee g_3 \vee \ldots\right)$$

$$g_1 \equiv \left(x_1 \neq x_2\right)$$
$$g_2 \equiv \left(x_2 \neq x_3\right)$$
$$\ldots$$

$$g_k \equiv f\left(x_{1,} \ldots, e\right)$$

- Any solution would have to include a variable from every clause, including those that encode equalities.
- In particular, it would have to include all the x_i

  → An exponential number of solutions has to be explored

$$\exists e \forall x_1 x_2 x_3 \ldots x_n \; \exists g_1 g_2 g_3 \ldots$$

$$(g_1 \vee g_2 \vee g_3 \vee \ldots)$$

$$g_1 \equiv (x_1 \neq x_2)$$

$$g_2 \equiv (x_2 \neq x_3)$$

$$\ldots$$

$$g_k \equiv f(x_{1,} \ldots, e)$$

- Any solution would have to include a variable from every clause, including those that encode equalities.
- In particular, it would have to include all the x_i

  → An exponential number of solutions has to be explored

In this simple example, a number of other techniques could work:

- Don't care propagation

$$\exists e \forall x_1 x_2 x_3 \dots x_n \; \exists g_1 g_2 g_3 \dots$$

$$\left(g_1 \vee g_2 \vee g_3 \vee \dots\right) \qquad g_1 \equiv \left(x_1 \neq x_2\right)$$

$$g_2 \equiv \left(x_2 \neq x_3\right) \qquad g_k \equiv f\left(x_{1,} \dots, e\right)$$

$$\dots$$

- Any solution would have to include a variable from every clause, including those that encode equalities.
- In particular, it would have to include all the x_i

  → An exponential number of solutions has to be explored

In this simple example, a number of other techniques could work:

- Don't care propagation
- Plaisted-Greenbaum encoding

$$\exists e \forall x_1 x_2 x_3 \ldots x_n \; \exists g_1 g_2 g_3 \ldots$$

$$(g_1 \vee g_2 \vee g_3 \vee \ldots)$$

$$g_1 \equiv (x_1 \neq x_2)$$

$$g_2 \equiv (x_2 \neq x_3)$$

$$\ldots$$

$$g_k \equiv f(x_{1,} \ldots, e)$$

• Any solution would have to include a variable from every clause, including those that encode equalities.
• In particular, it would have to include all the x_i

→ An exponential number of solutions has to be explored

In this simple example, a number of other techniques could work:
• Don't care propagation
• Plaisted-Greenbaum encoding

$$\exists e \forall x_1 x_2 x_3 \ldots x_n \; \exists g_1 g_2 g_3 \ldots$$

$$(g_1 \vee g_2 \vee g_3 \vee \ldots)$$

$$g_1 \Rightarrow (x_1 \neq x_2)$$

$$g_2 \Rightarrow (x_2 \neq x_3)$$

$$\ldots$$

$$g_k \Rightarrow f(x_{1,} \ldots, e)$$

- Any solution would have to include a variable from every clause, including those that encode equalities.
- In particular, it would have to include all the x_i

  → An exponential number of solutions has to be explored

In this simple example, a number of other techniques could work:

- Don't care propagation
- Plaisted-Greenbaum encoding

May get a solution

$$(x_1 \wedge \neg x_2)$$

$$\exists e \forall x_1 x_2 x_3 \ldots x_n \; \exists g_1 g_2 g_3 \ldots$$

$$(g_1 \vee g_2 \vee g_3 \vee \ldots)$$

$$g_1 \Rightarrow (x_1 \neq x_2)$$

$$g_2 \Rightarrow (x_2 \neq x_3)$$

$$\ldots$$

$$g_k \Rightarrow f(x_{1,} \ldots, e)$$

- Any solution would have to include a variable from every clause, including those that encode equalities.
- In particular, it would have to include all the x_i

  → An exponential number of solutions has to be explored

In this simple example, a number of other techniques could work:

- Don't care propagation
- Plaisted-Greenbaum encoding

BUT NOT IN GENERAL

Example:

$$\exists e \forall x_1 x_2 x_3 \ldots x_n$$
$$(x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \ldots \oplus x_n) \vee f(x_{1,} x_{2,} x_{3,} x_{4,} e)$$

- Main problem: cubes are not expressive enough to represent more than one solution to

$$\left(x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \ldots \oplus x_n\right)$$

BUT NOT IN GENERAL

Example:

$$\exists e \forall x_1 x_2 x_3 \ldots x_n$$

$$\left(x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \ldots \oplus x_n\right) \vee f\left(x_{1,} x_{2,} x_{3,} x_{4,} e\right)$$

- Main problem: cubes are not expressive enough to represent more than one solution to

$$\left(x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \ldots \oplus x_n\right)$$

- Such is not a problem for conflicts, because we have Tseitin variables

BUT NOT IN GENERAL

Example:

$$\exists e \forall x_1 x_2 x_3 \ldots x_n \; \exists g_1 g_2 g_3 \ldots g_k$$
$$\left(x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \ldots \oplus x_n\right) \vee f\left(x_{1,} x_{2,} x_{3,} x_{4,} e\right)$$

- Main problem: cubes are not expressive enough to represent more than one solution to

$$\left(x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \ldots \oplus x_n\right)$$

- Such is not a problem for conflicts, because we have Tseitin variables
- But they are not useful in cubes

BUT NOT IN GENERAL

Example:

$$\exists e \forall x_1 x_2 x_3 \ldots x_n \; \exists g_1 g_2 g_3 \ldots g_k$$

$$\left(x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \ldots \oplus x_n\right) \vee f\left(x_1, x_2, x_3, x_4, e\right)$$

- Main problem: cubes are not expressive enough to represent more than one solution to

$$\left(x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \ldots \oplus x_n\right)$$

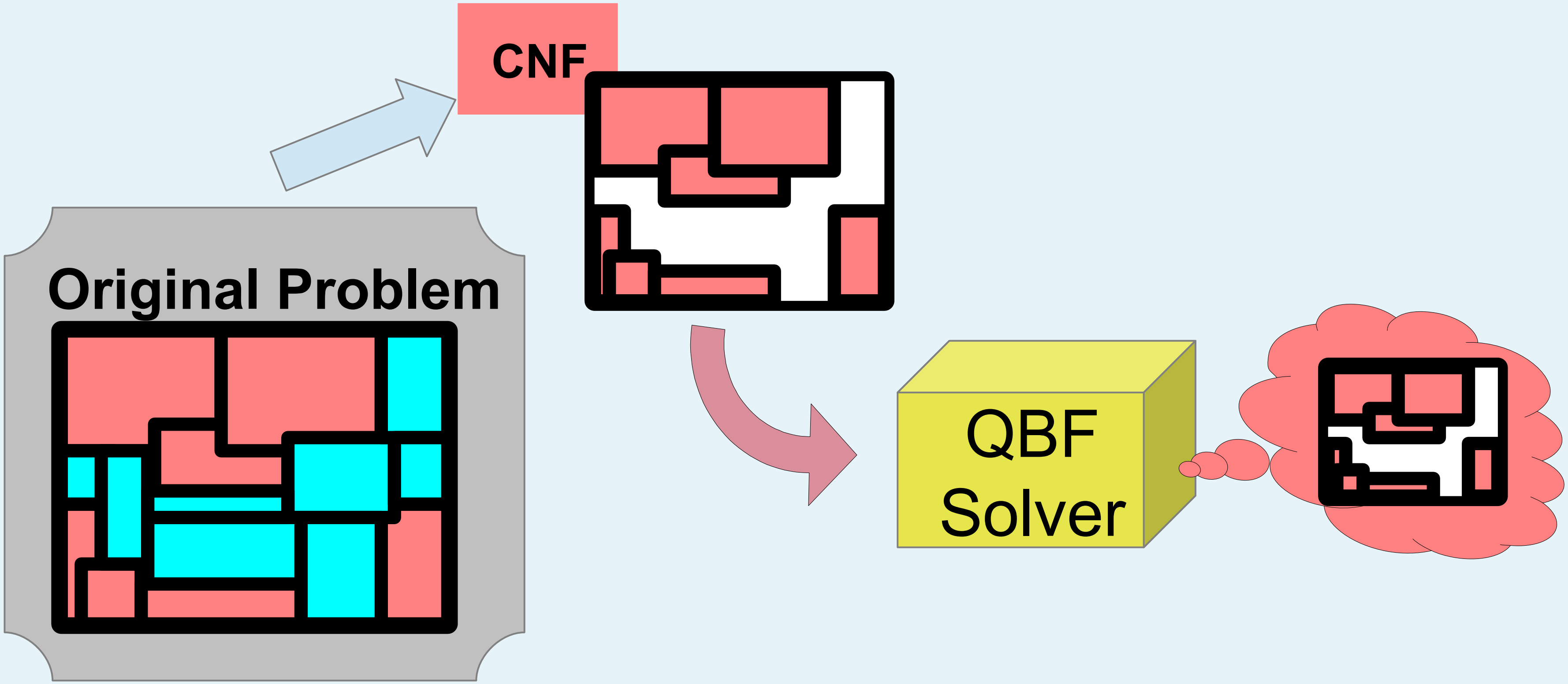
- Such is not a problem for conflicts, because we have Tseitin variables
- But they are not useful in cubes
- So: we lose the ability to generalize solutions

BUT NOT IN GENERAL

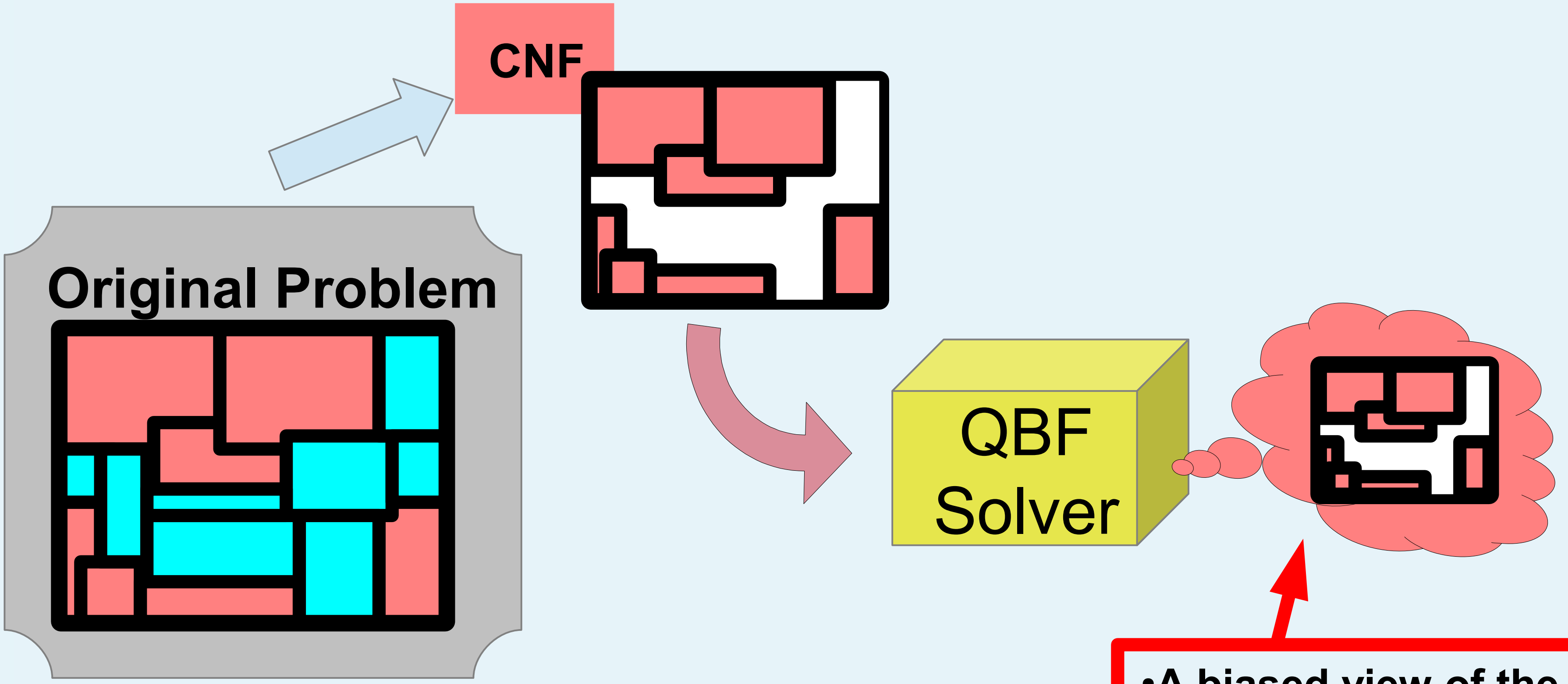
Example:

$$\exists e \forall x_1 x_2 x_3 \ldots x_n \ \exists g_1 g_2 g_3 \ldots g_k$$
$$\left(x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \ldots \oplus x_n\right) \vee f\left(x_{1,} x_{2,} x_{3,} x_{4,} e\right)$$

QBF Solving



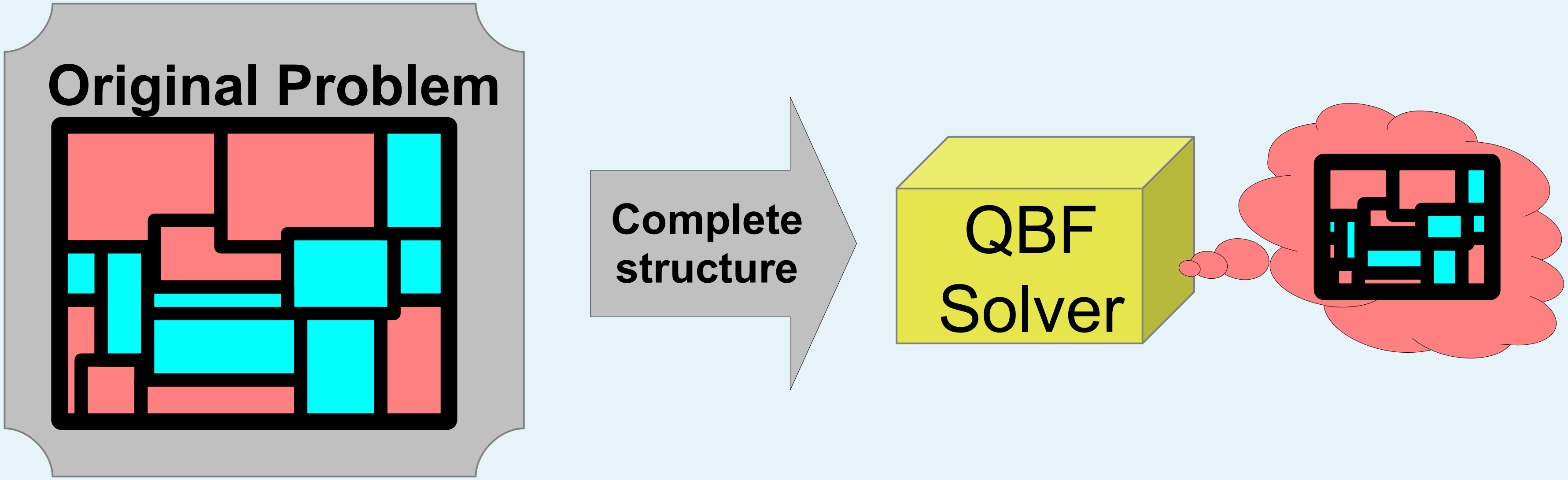
QBF Solving



- A biased view of the problem

QBF Solving

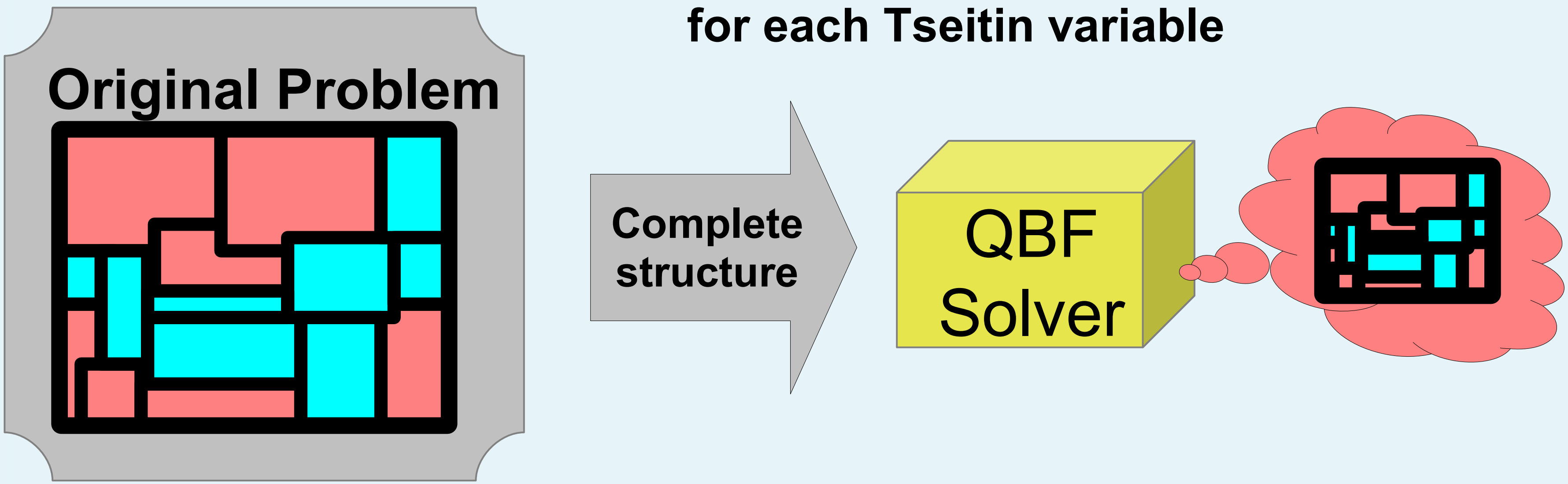
One solution: non-CNF



QBF Solving

One solution: non-CNF

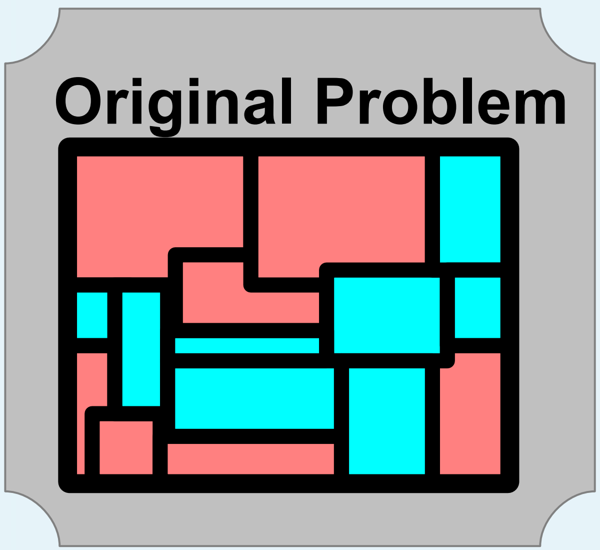
- Implicitly creates a universal copy for each Tseitin variable

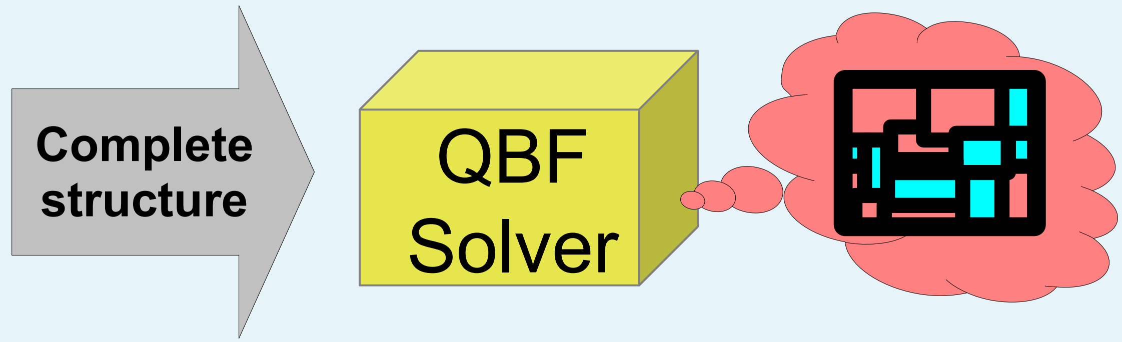


QBF Solving

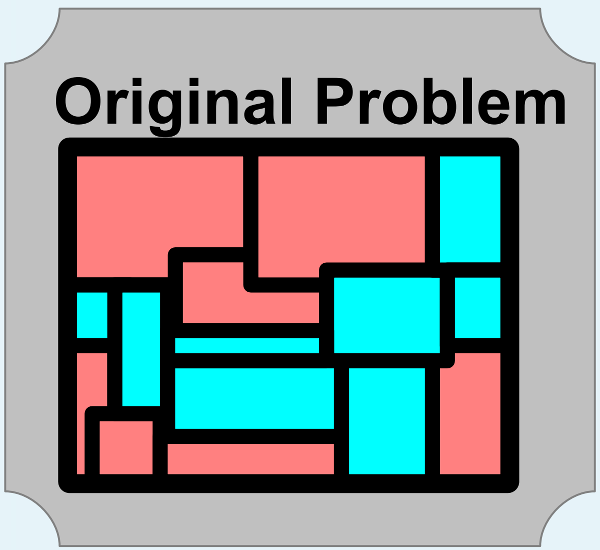
One solution: non-CNF

- Loses implementation efficiency
- Needs specialized techniques
- Limited benefits from other work

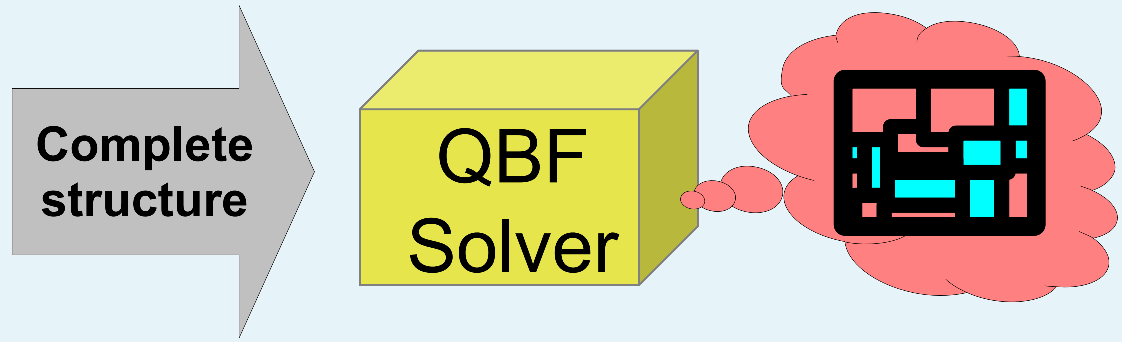




QBF Solving

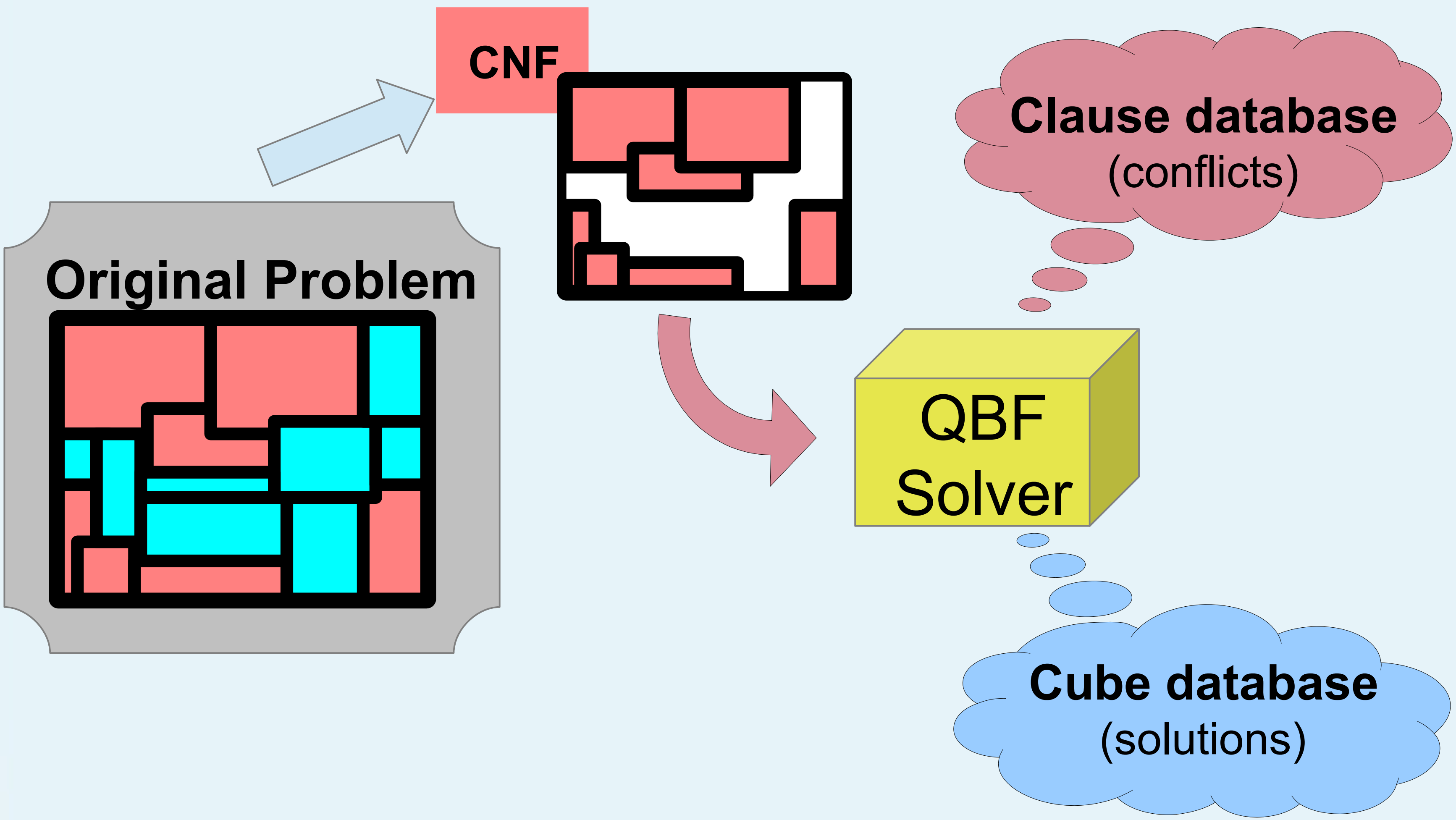


One solution: non-CNF
- **Loses implementation efficiency**
- **Needs specialized techniques**
- **Limited benefits from other work**

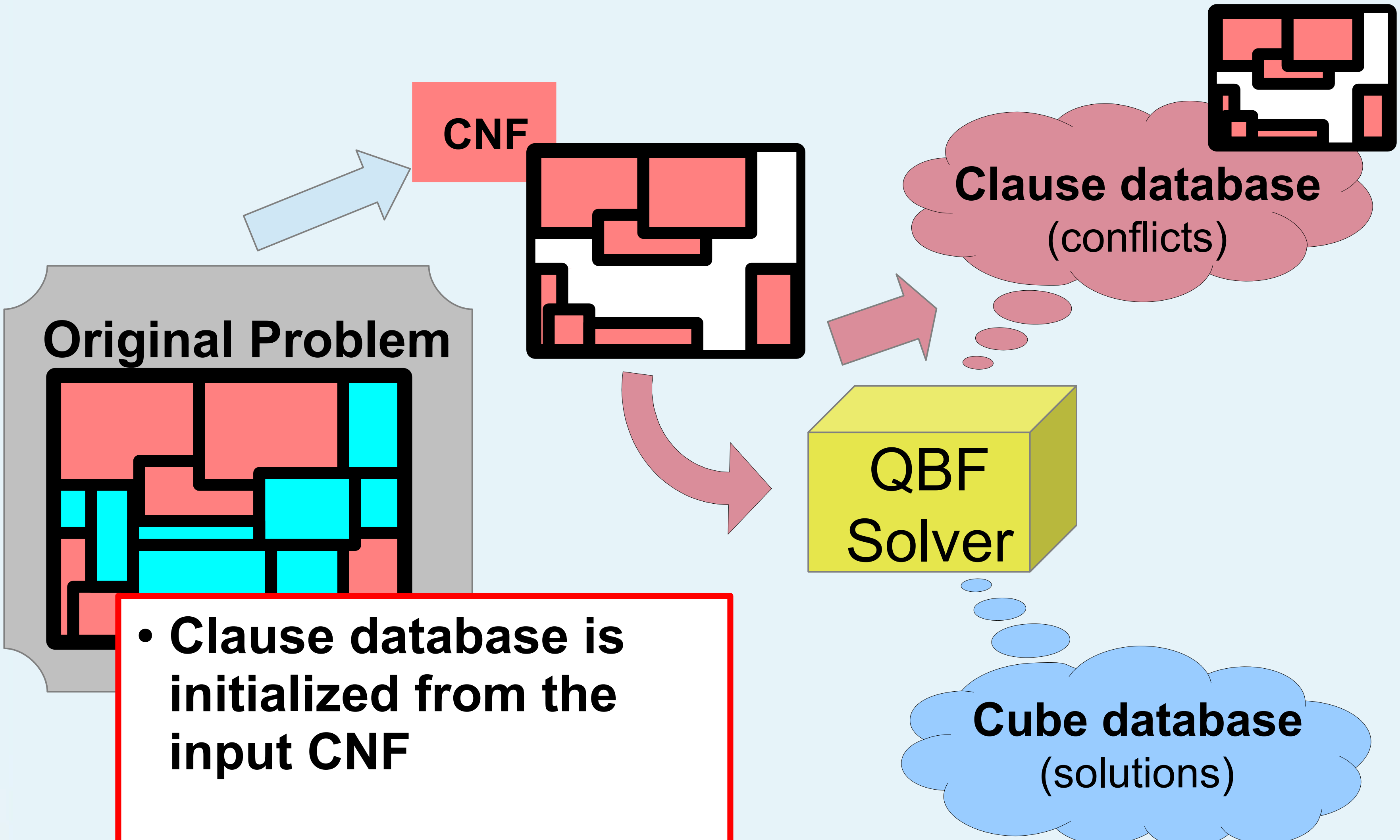


Specialized solvers are unnecessary.

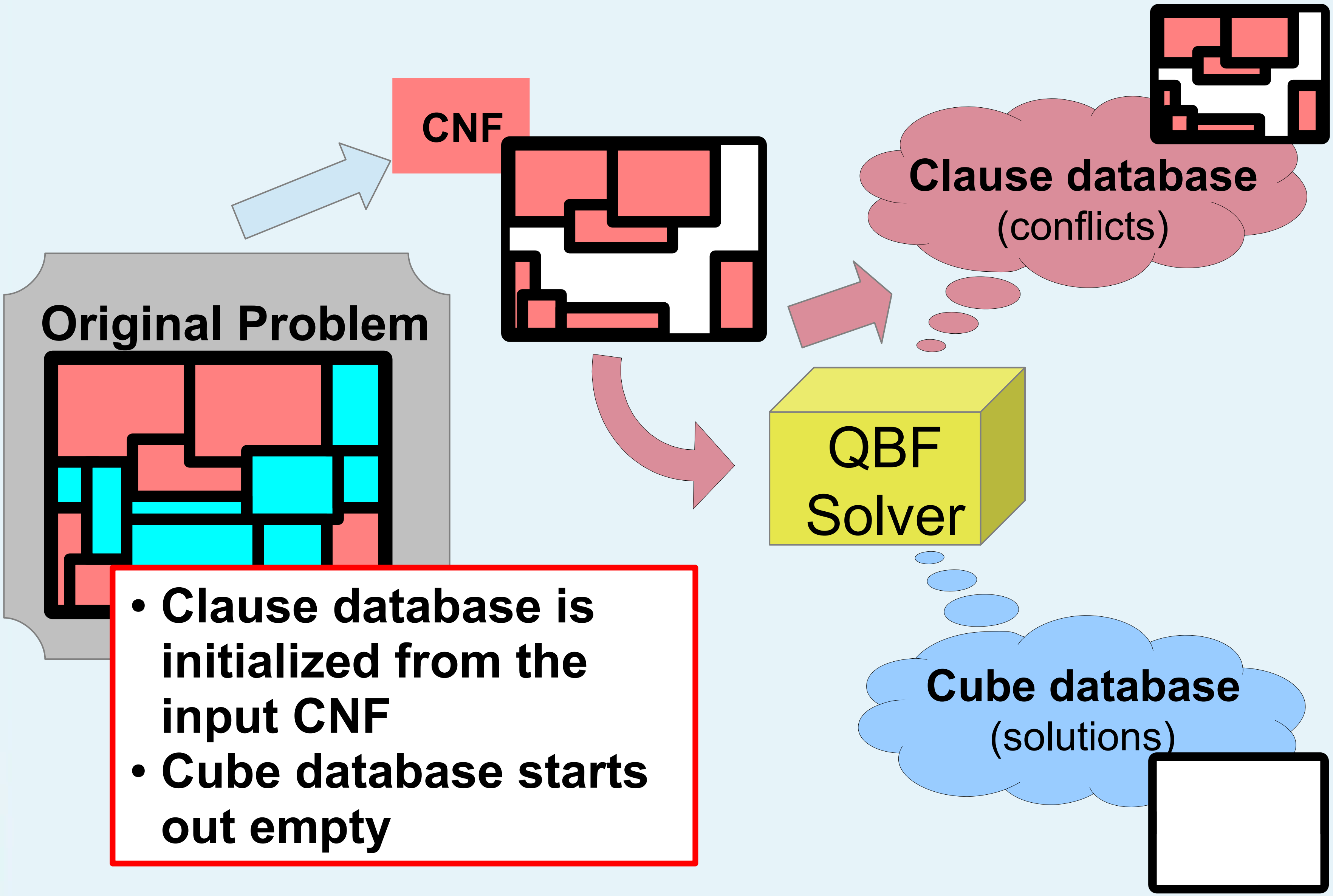
Existing search-based solvers have all the needed mechanisms



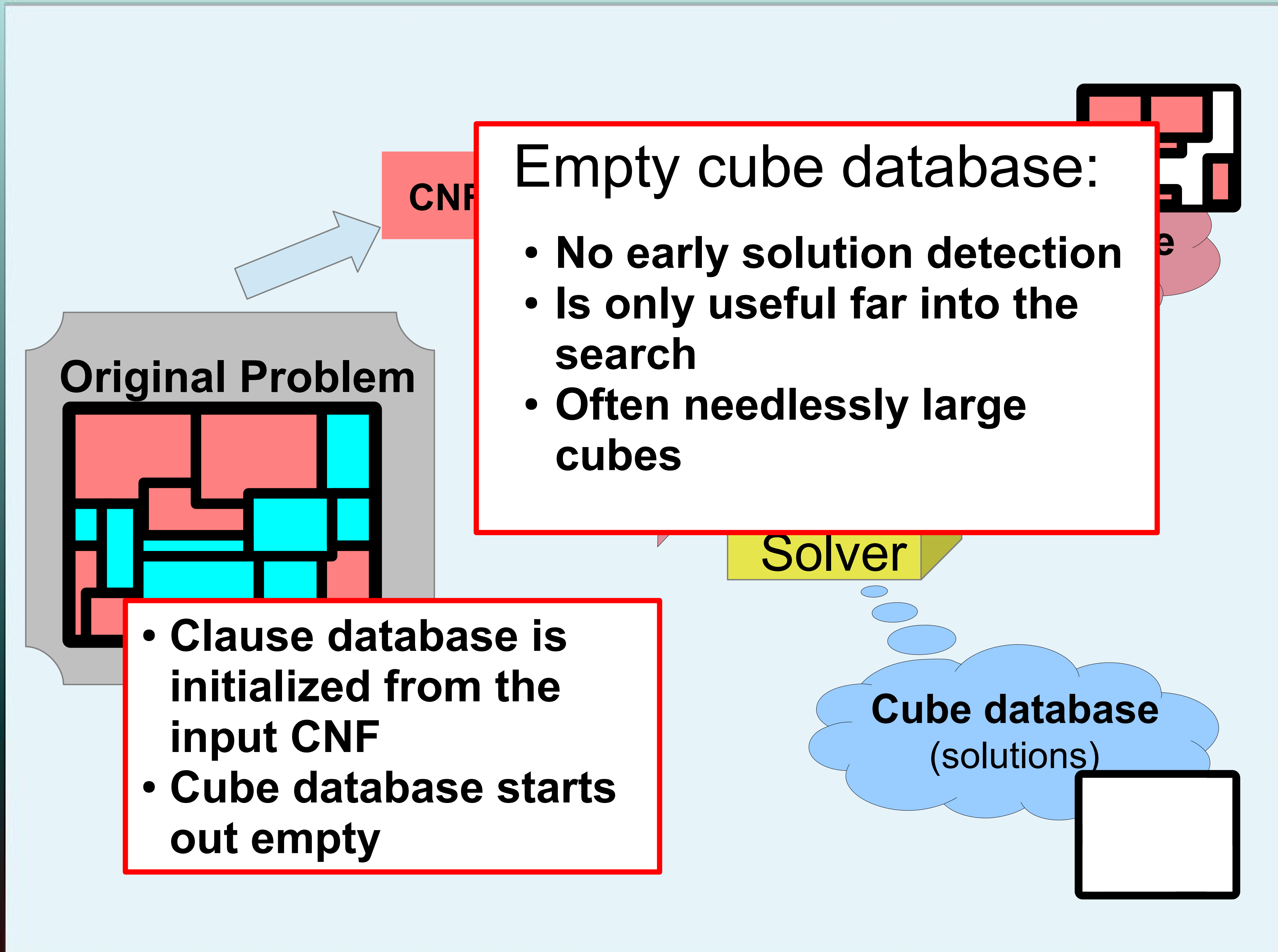
CNF
Original Problem
Clause database
(conflicts)
QBF
Solver
Cube database
(solutions)

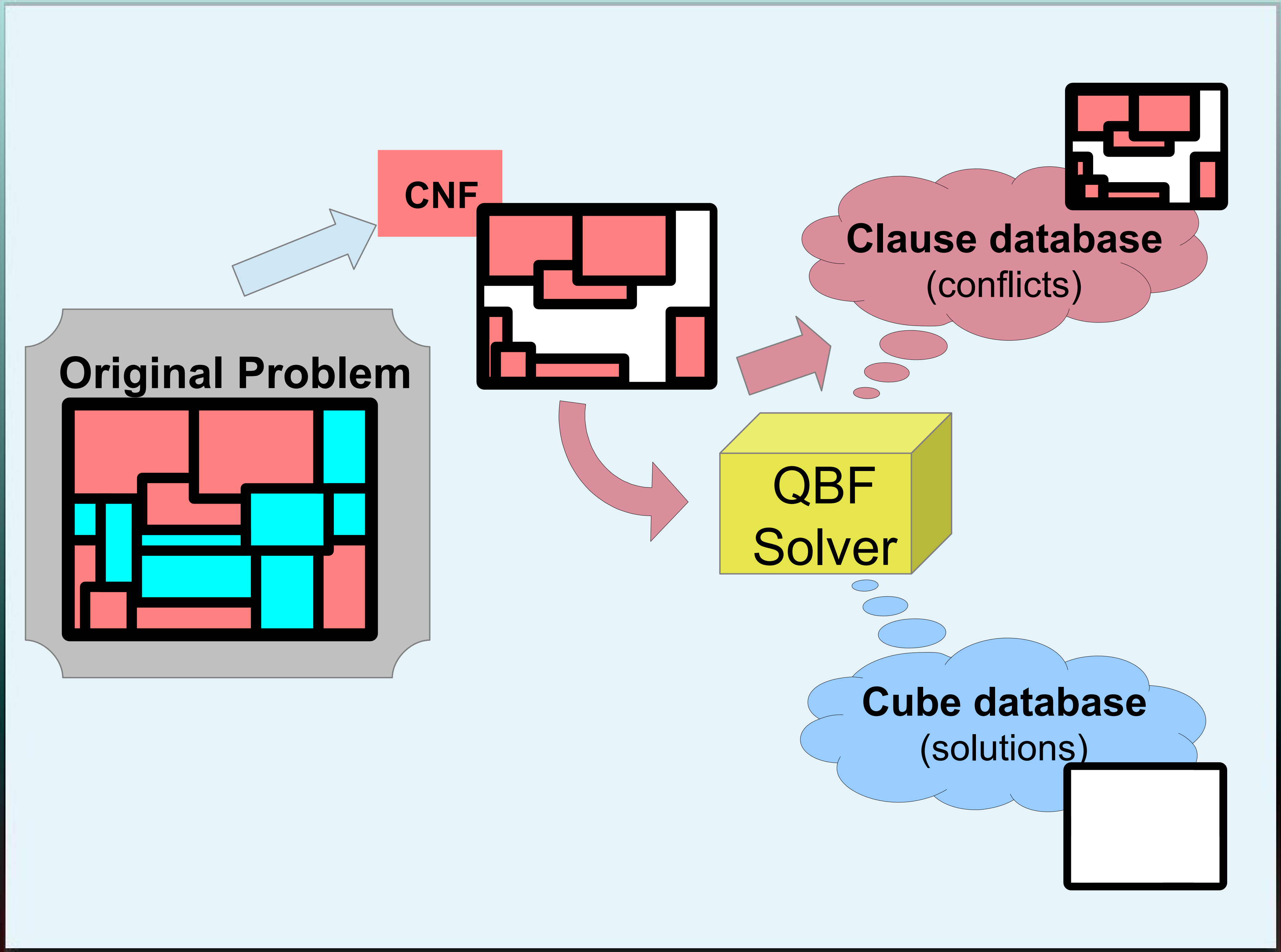


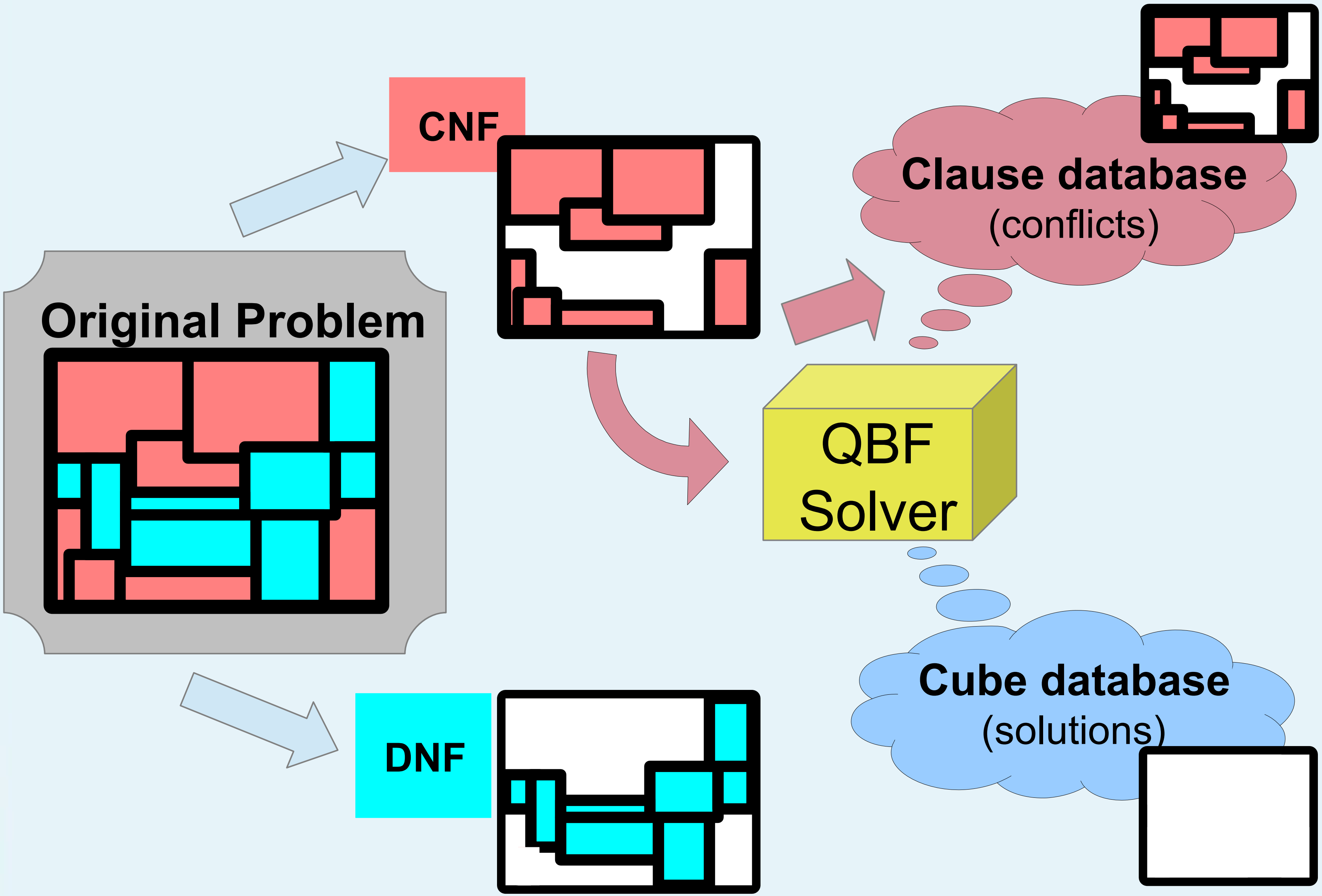
- Clause database is initialized from the input CNF



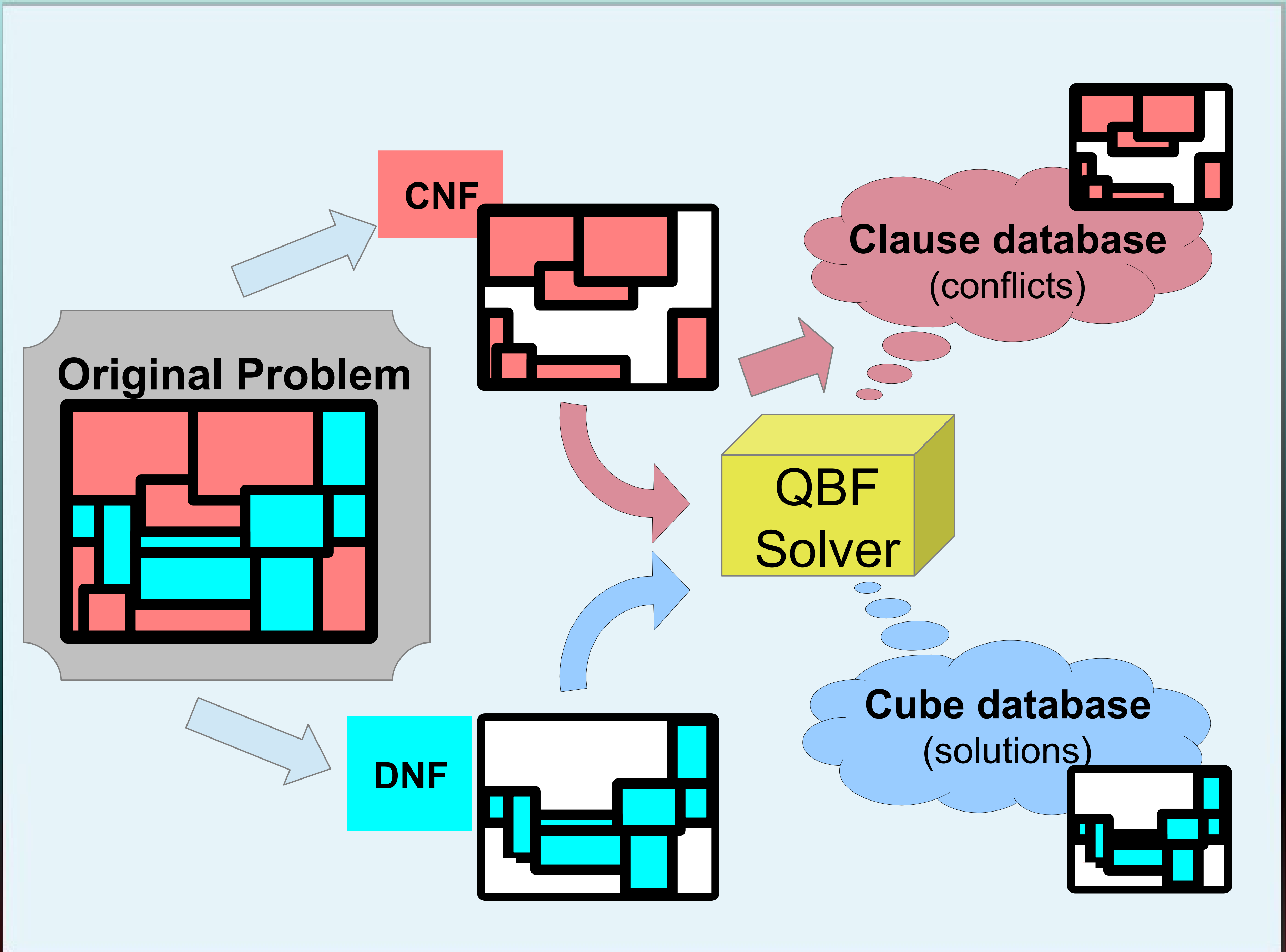
- Clause database is initialized from the input CNF
- Cube database starts out empty


Empty cube database:
• No early solution detection
• Is only useful far into the search
• Often needlessly large cubes
CNF
Original Problem
Solver
Cube database
(solutions)
• Clause database is initialized from the input CNF
• Cube database starts out empty

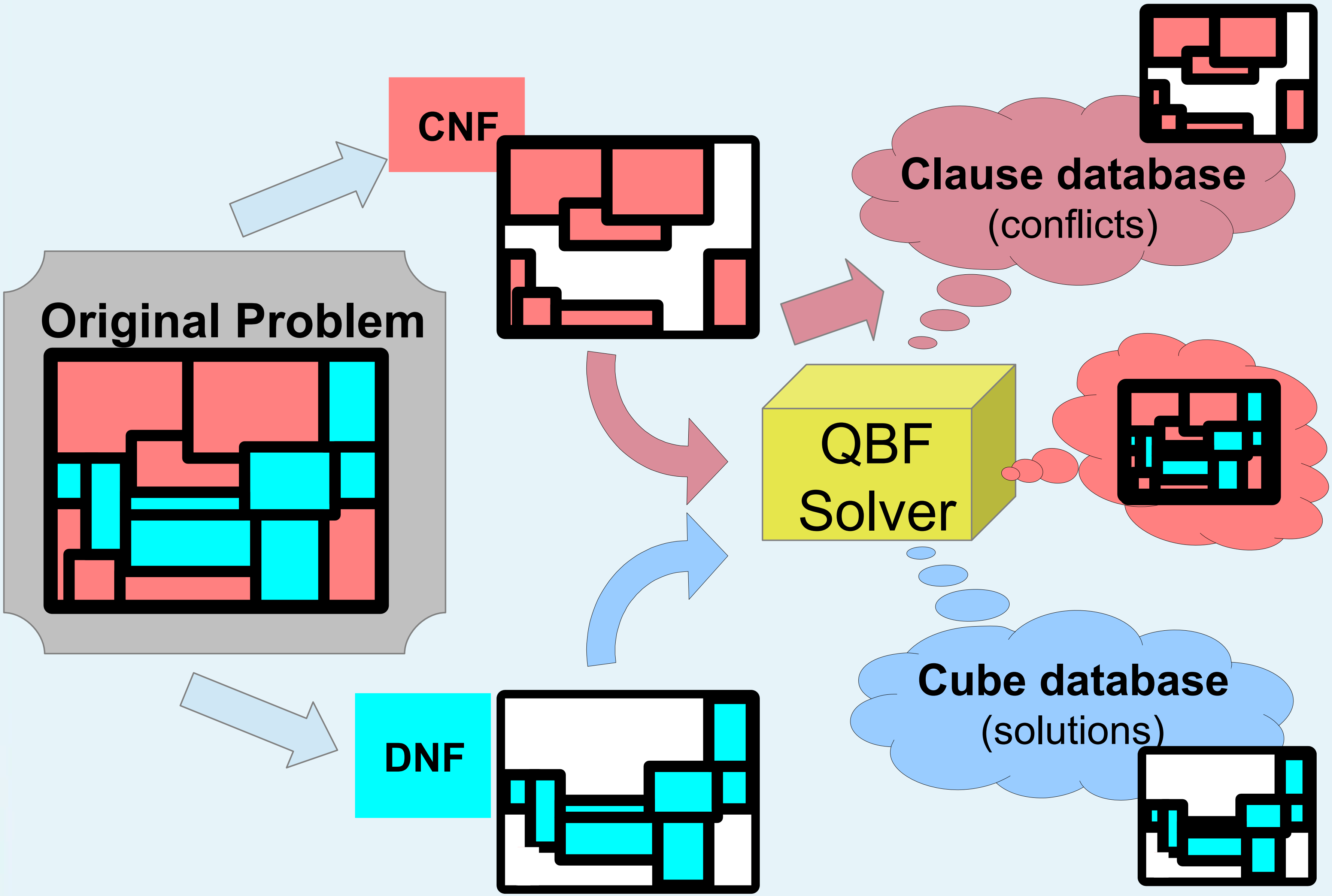

CNF
Clause database
(conflicts)
Original Problem
QBF
Solver
Cube database
(solutions)



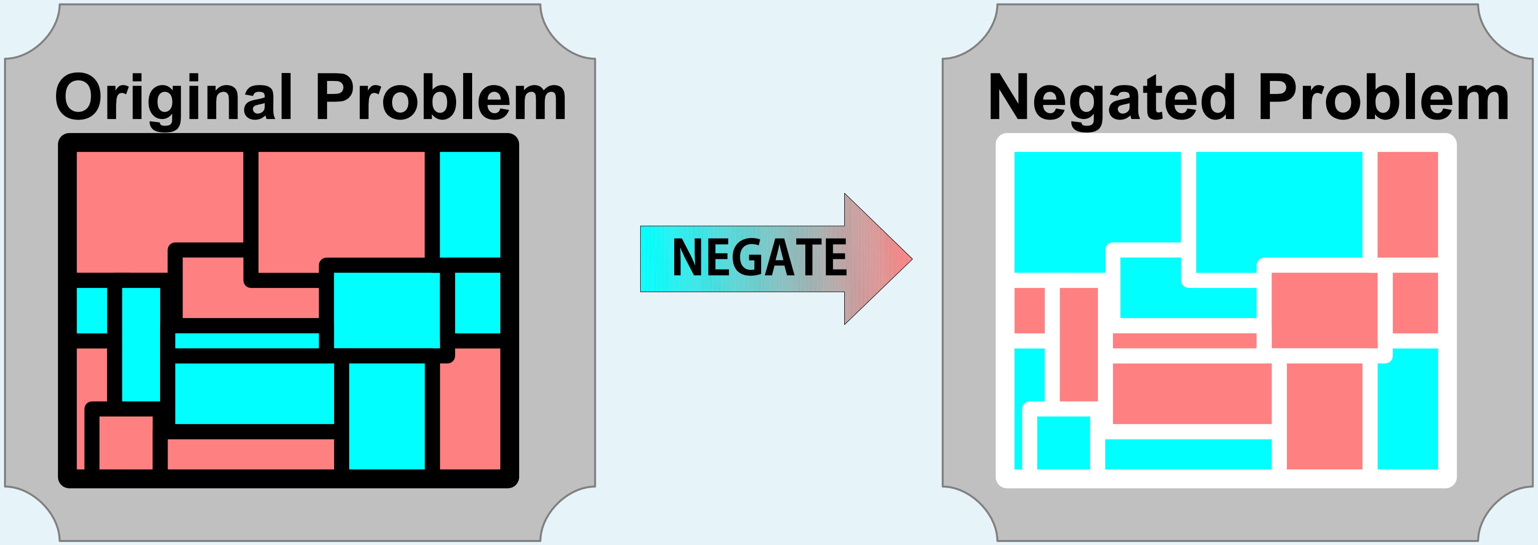
Original Problem
CNF
DNF
QBF Solver
Clause database
(conflicts)
Cube database
(solutions)



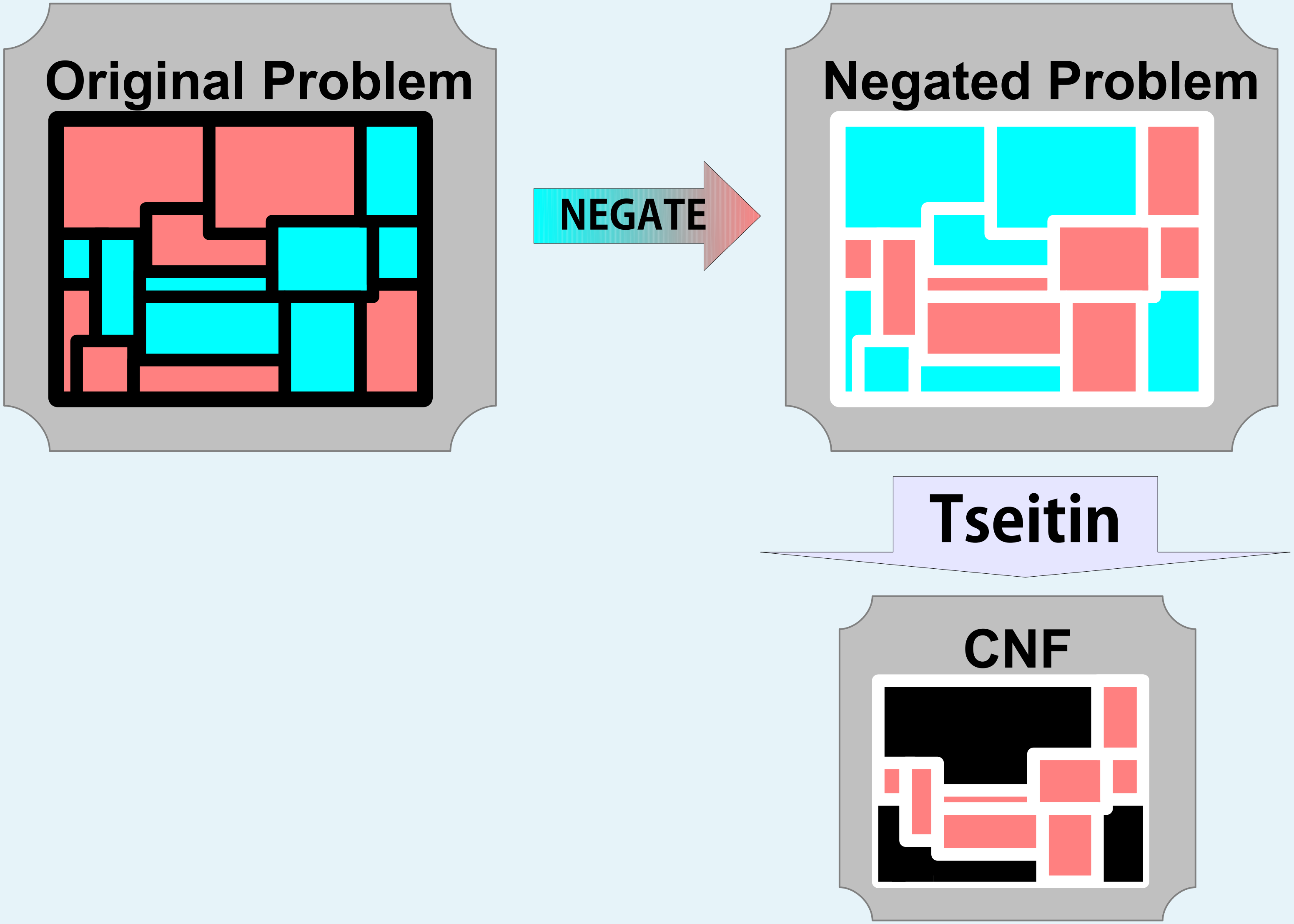
CNF
Clause database
(conflicts)
Original Problem
QBF
Solver
DNF
Cube database
(solutions)


CNF
Clause database
(conflicts)
Original Problem
QBF
Solver
DNF
Cube database
(solutions)

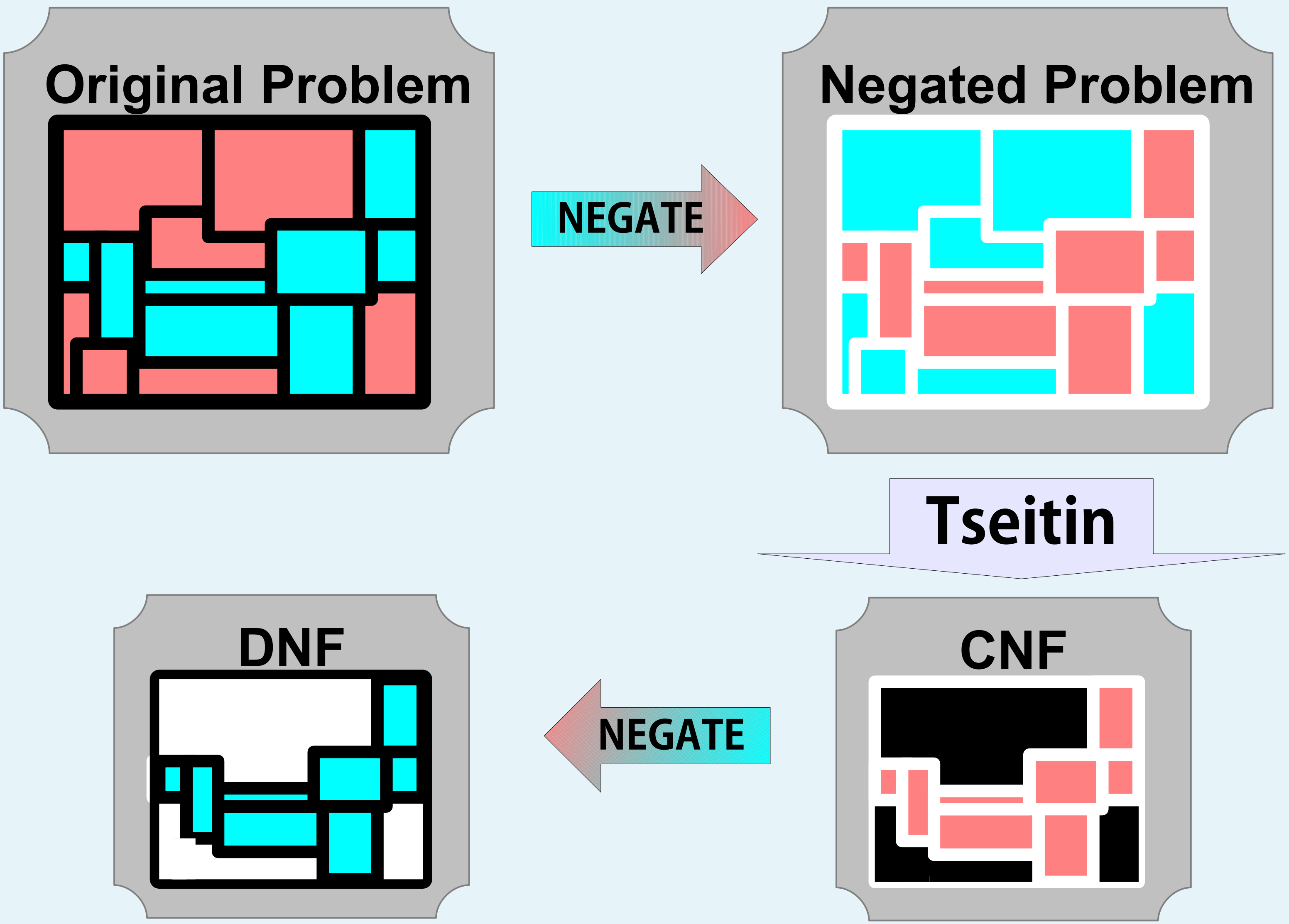
Getting solution structure



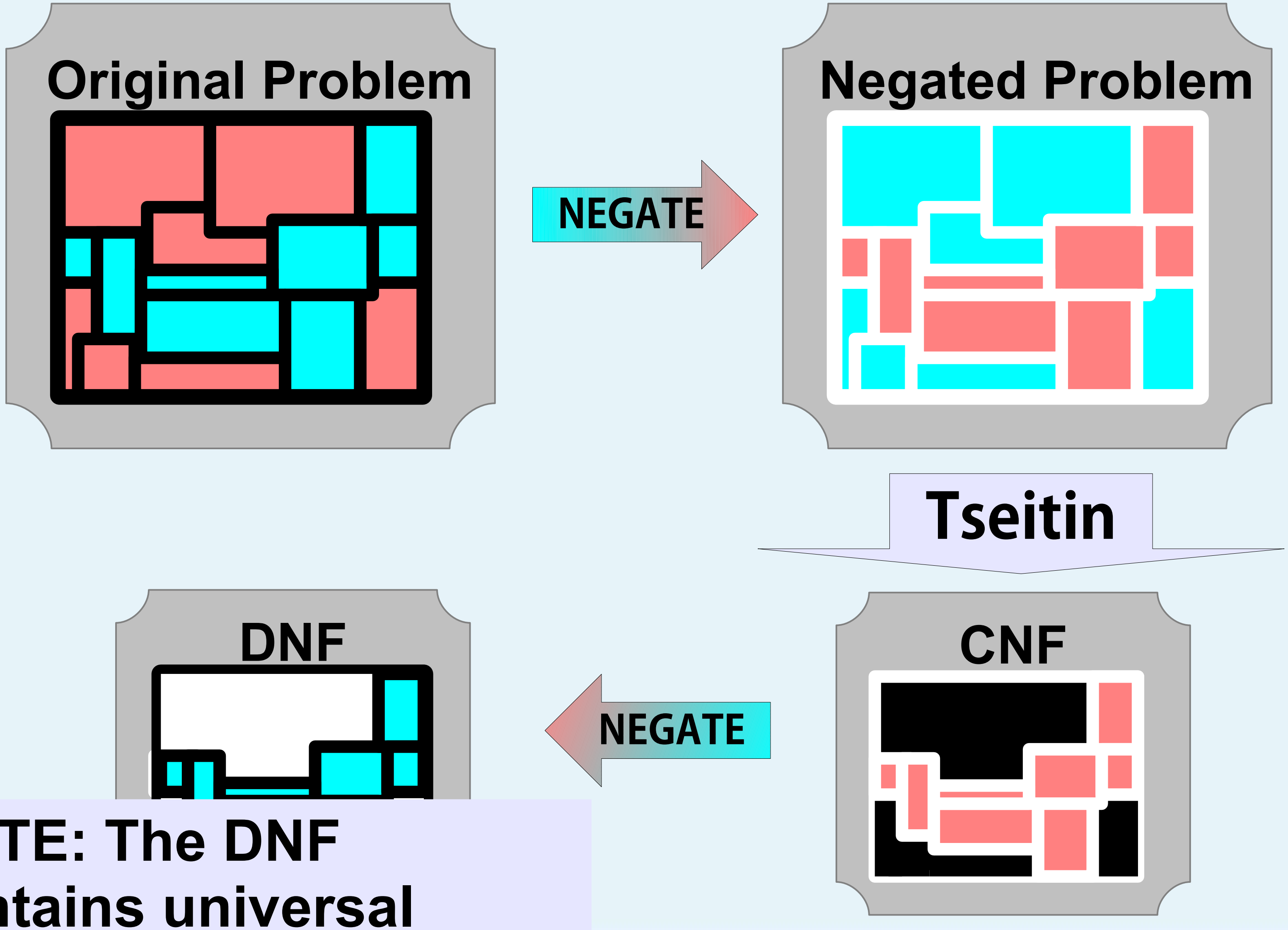
Getting solution structure



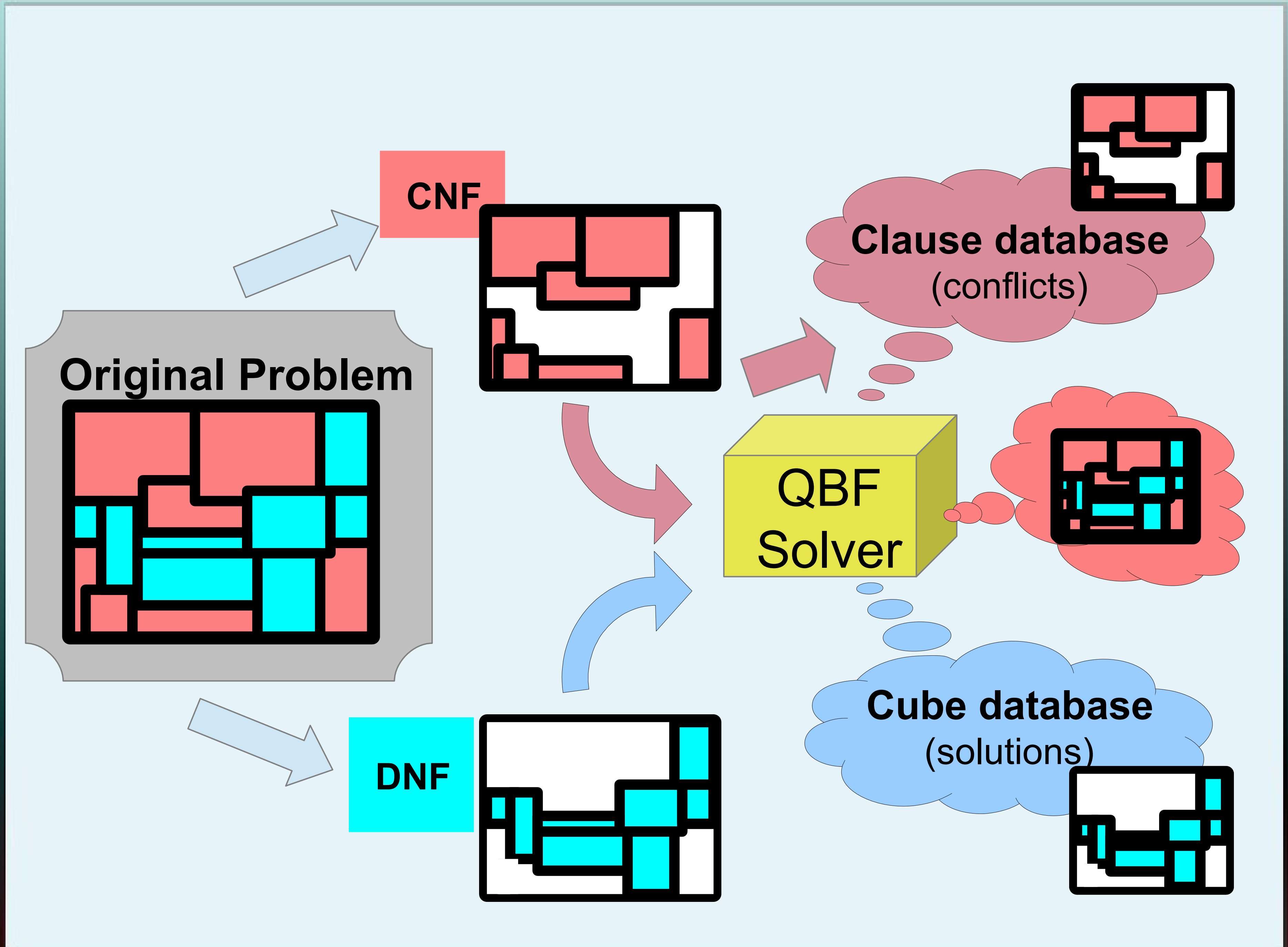
Getting solution structure

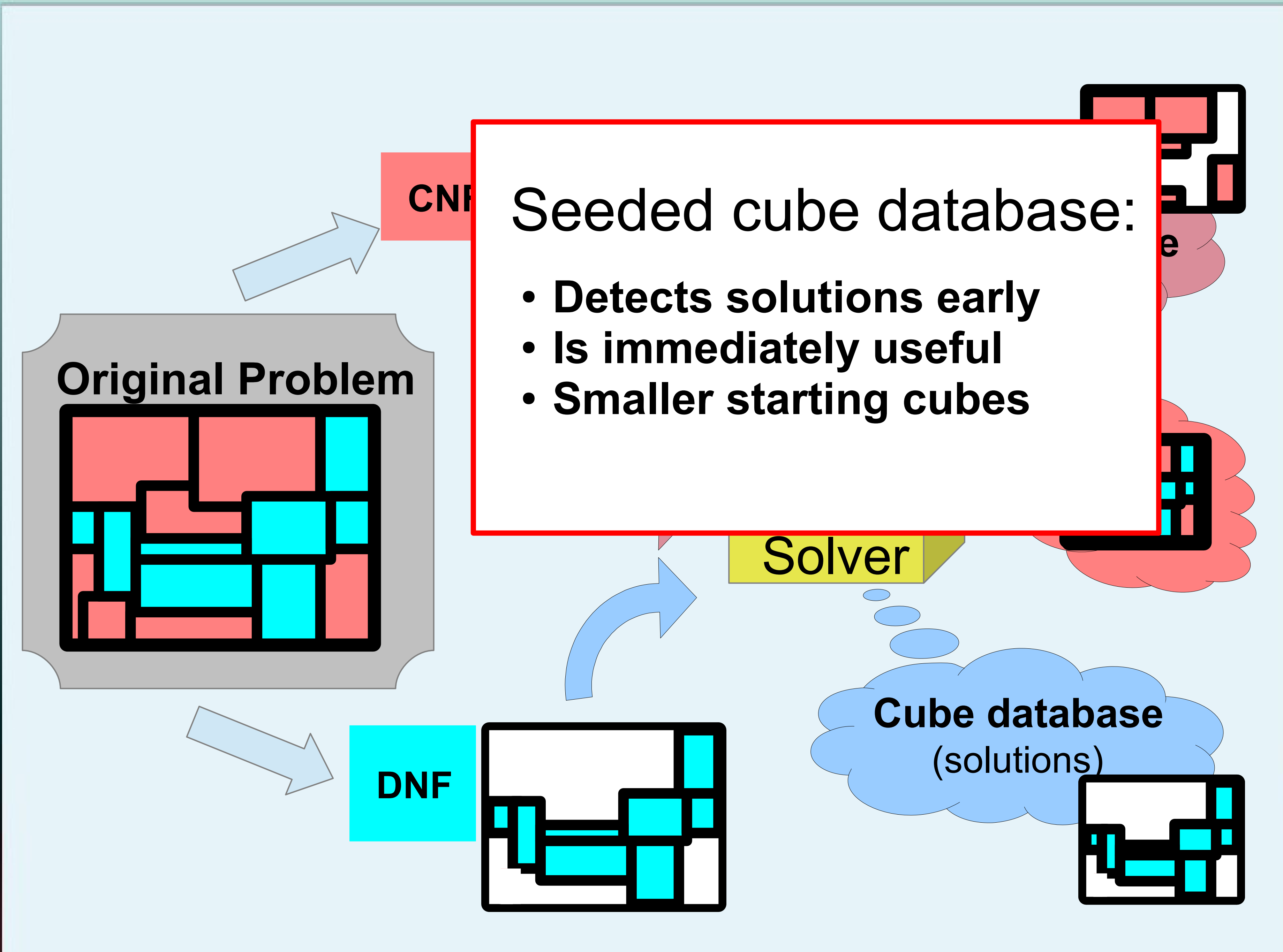


Getting solution structure



NOTE: The DNF contains universal Tseitin variables


CNF
Clause database
(conflicts)
Original Problem
QBF
Solver
DNF
Cube database
(solutions)



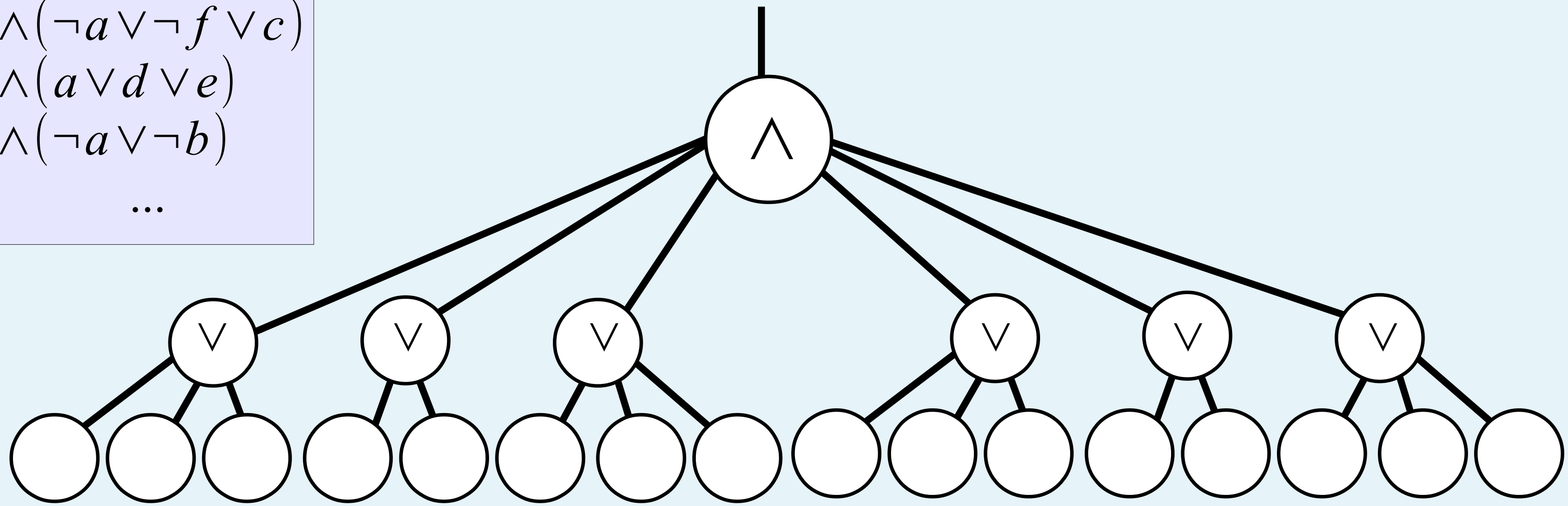
Seeded cube database:
• Detects solutions early
• Is immediately useful
• Smaller starting cubes
Original Problem
DNF
Solver
Cube database
(solutions)

Reconstructing structure

- Problem: original non-CNF is not always available
- Reconstruction methods exist, but they are necessarily unreliable and incomplete
- Want to take advantage of partially reconstructed information

Reconstructing structure

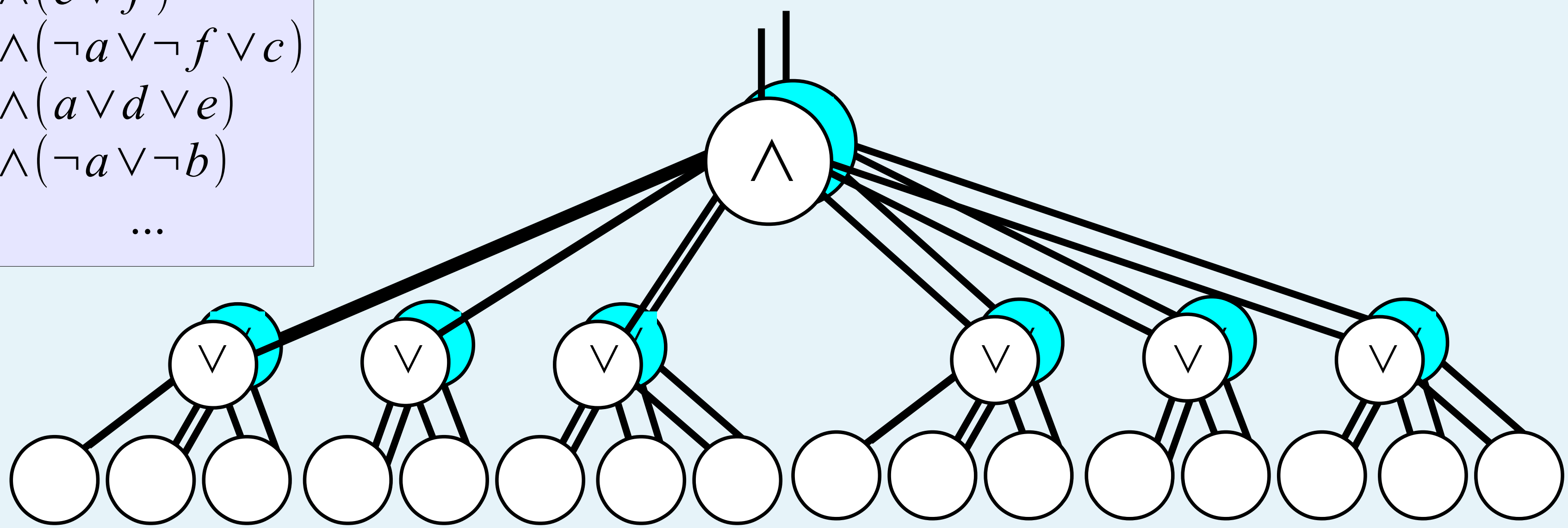
$$
\begin{aligned}
&(a \vee b \vee c) \\
\wedge &(e \vee f) \\
\wedge &(\neg a \vee \neg f \vee c) \\
\wedge &(a \vee d \vee e) \\
\wedge &(\neg a \vee \neg b) \\
&\ldots
\end{aligned}
$$



- **CNF can be viewed as a flat tree**

Reconstructing structure

$$
\begin{aligned}
&(a \vee b \vee c) \\
\wedge &(e \vee f) \\
\wedge &(\neg a \vee \neg f \vee c) \\
\wedge &(a \vee d \vee e) \\
\wedge &(\neg a \vee \neg b) \\
&\ldots
\end{aligned}
$$

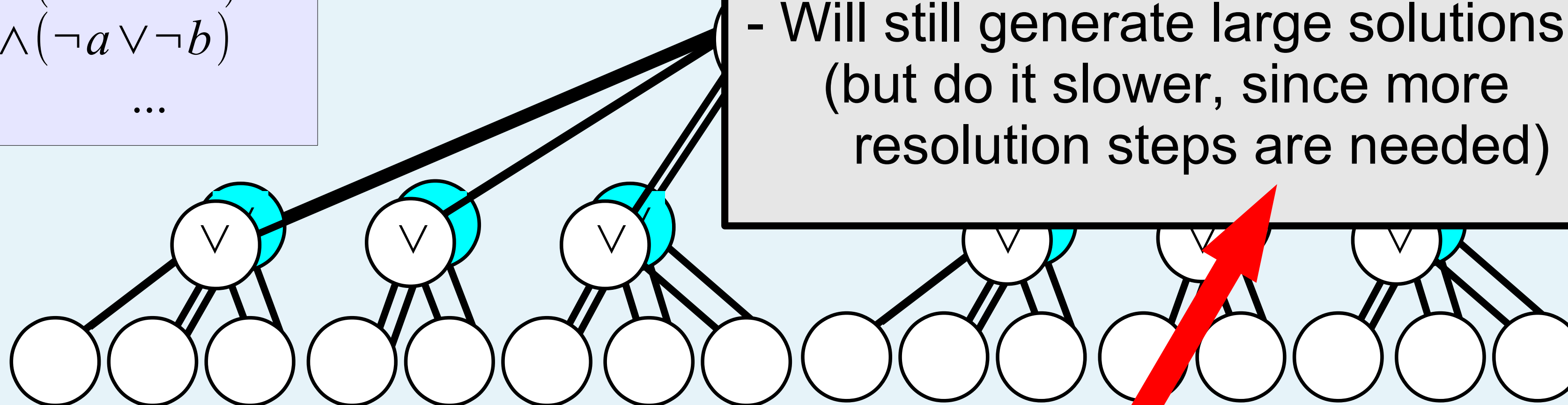


- **CNF can be viewed as a flat tree**
- **Negating it and converting to DNF would create a new variable for every clause**

Reconstructing structure

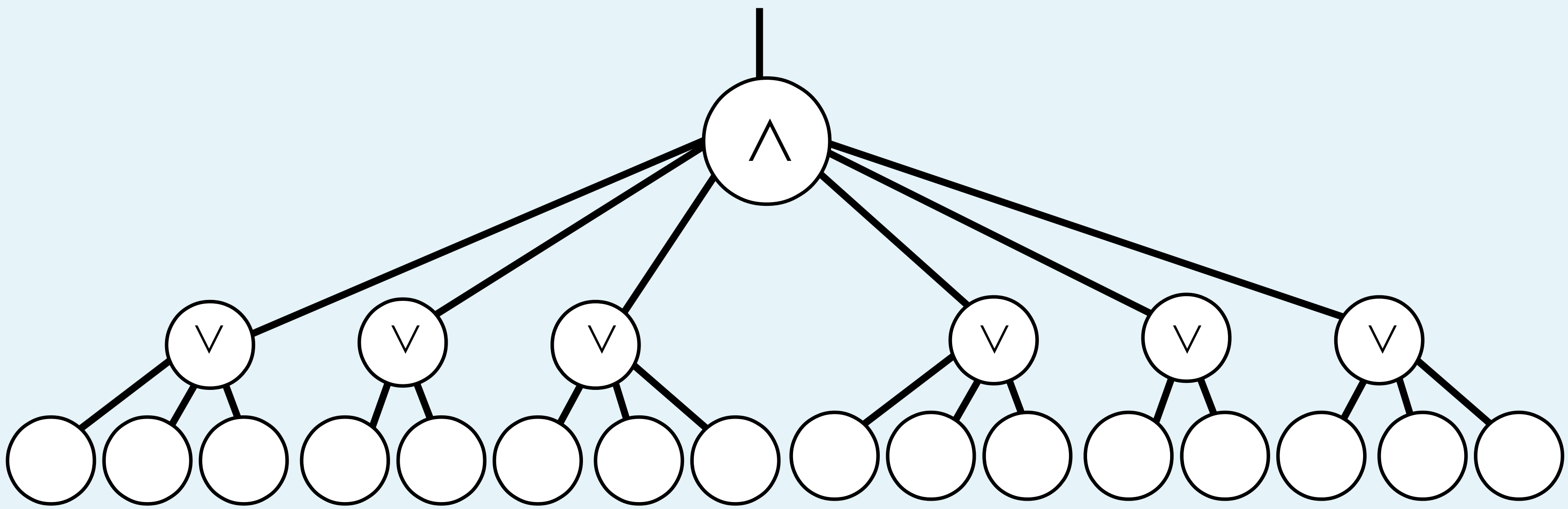
$$
\begin{aligned}
&(a \vee b \vee c) \\
&\wedge (e \vee f) \\
&\wedge (\neg a \vee \neg f \vee c) \\
&\wedge (a \vee d \vee e) \\
&\wedge (\neg a \vee \neg b) \\
&\quad \ldots
\end{aligned}
$$

Is not generally useful:
- Will not help early detection
- Will not help propagation
- Will still generate large solutions
  (but do it slower, since more
  resolution steps are needed)

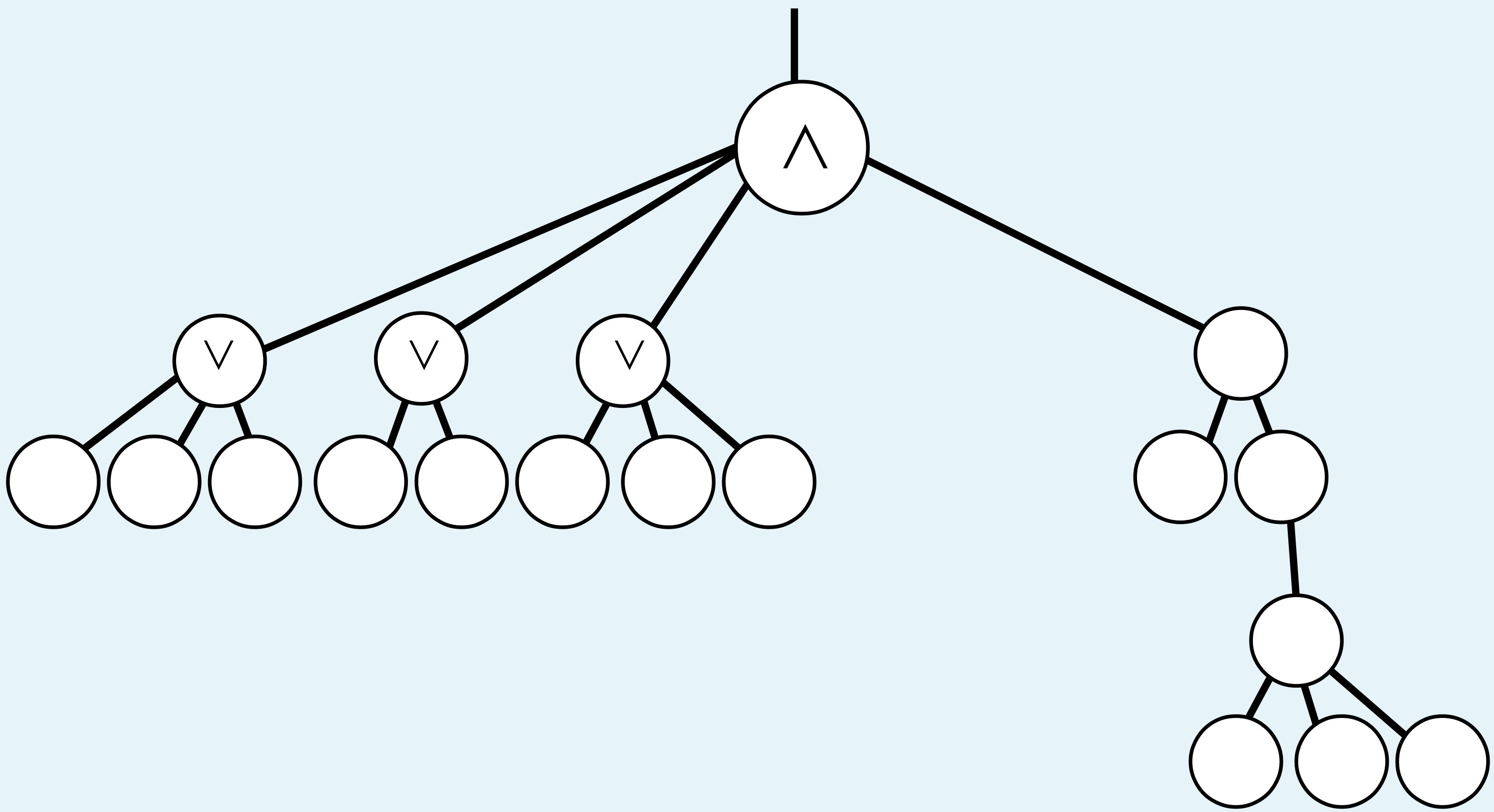


- **CNF can be viewed as a flat tree**
- **Negating it and converting to DNF would create a new variable for every clause**

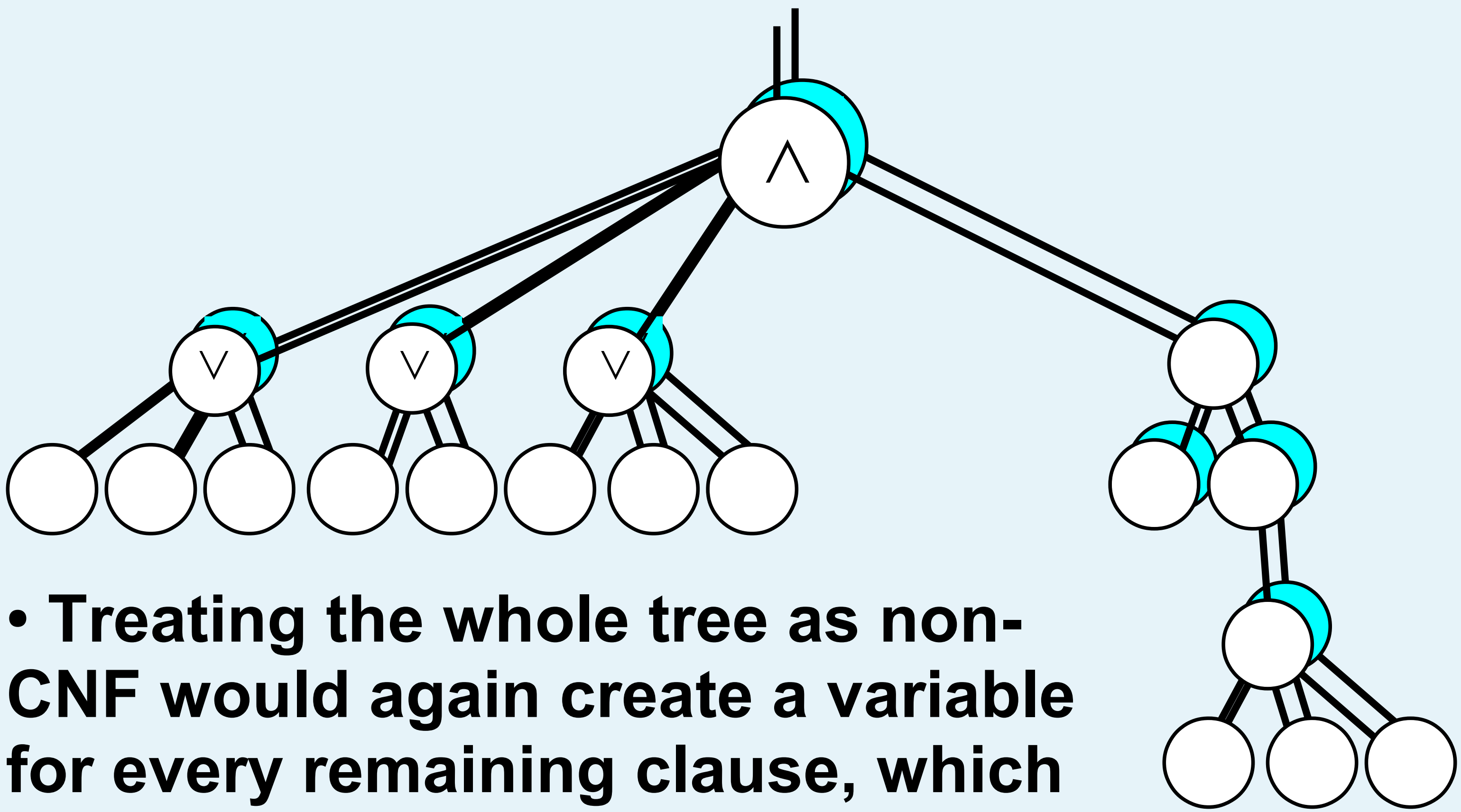
Reconstructing structure



Reconstructing structure

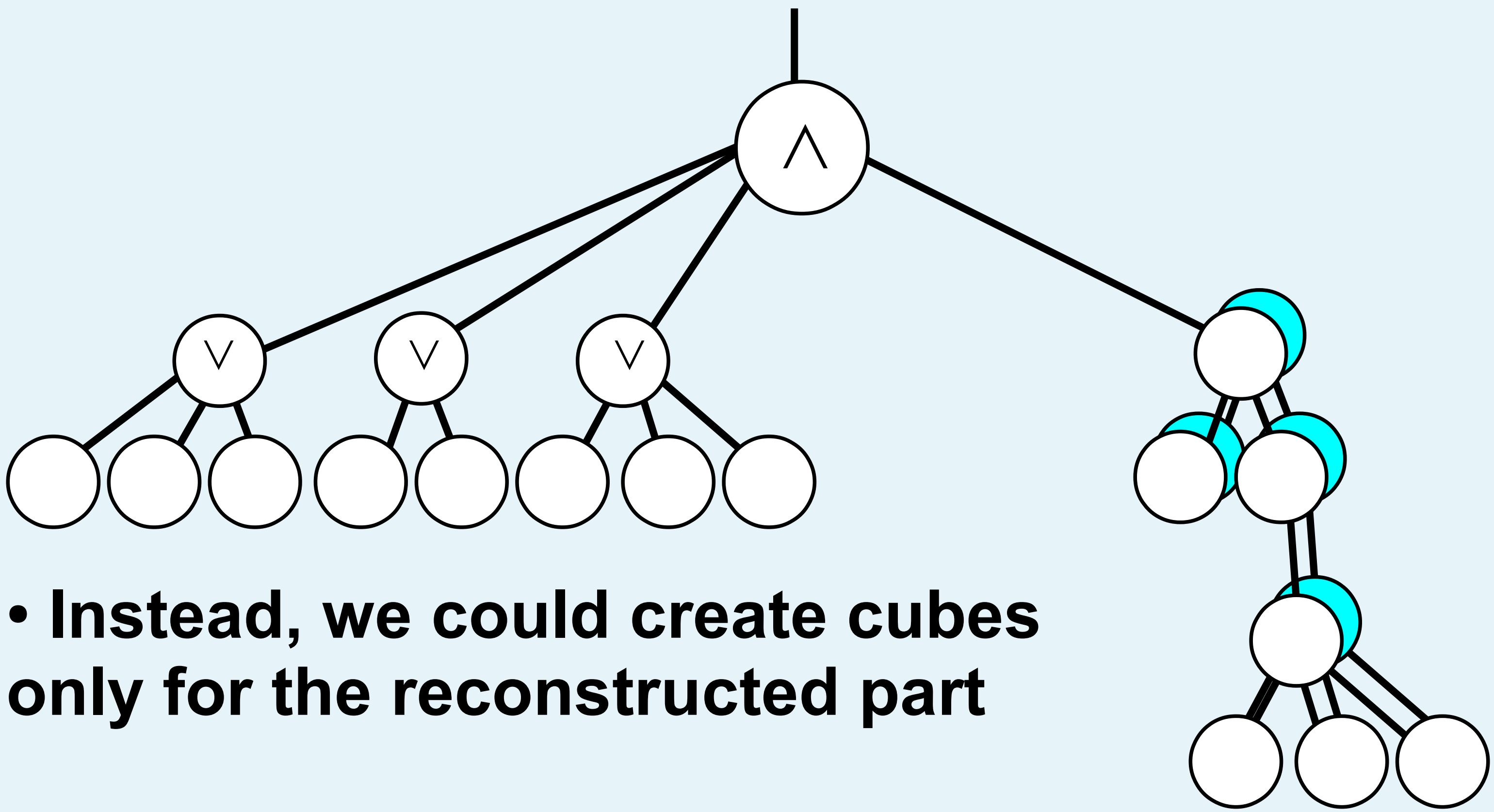


Reconstructing structure



- **Treating the whole tree as non-CNF would again create a variable for every remaining clause, which is inefficient**

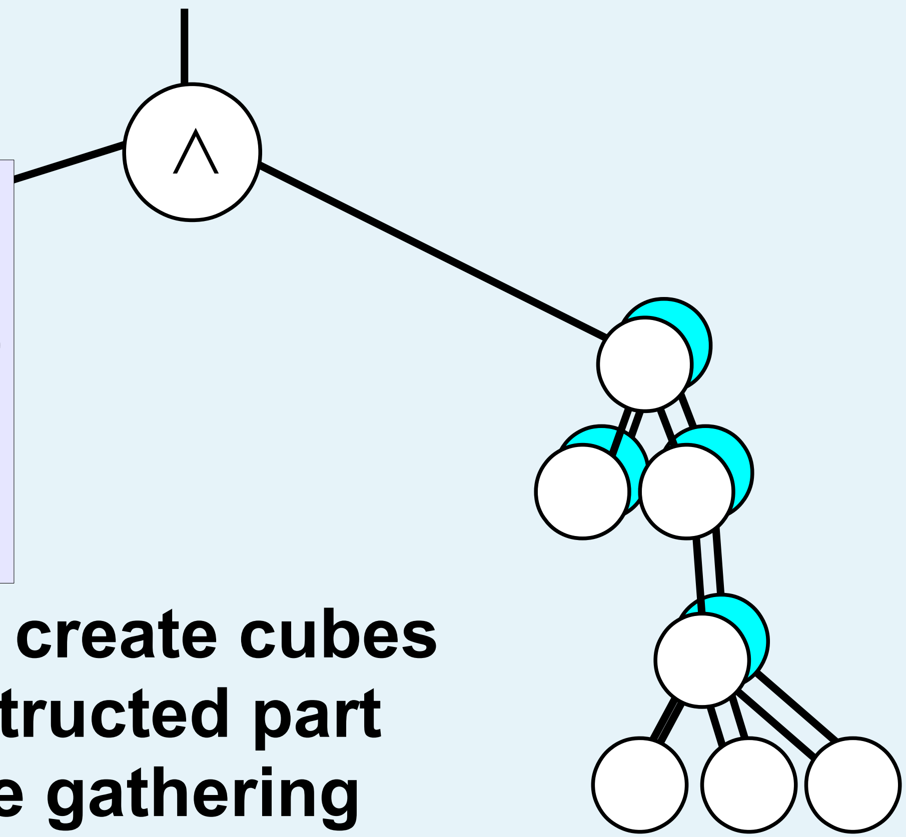
Reconstructing structure



- **Instead, we could create cubes only for the reconstructed part**

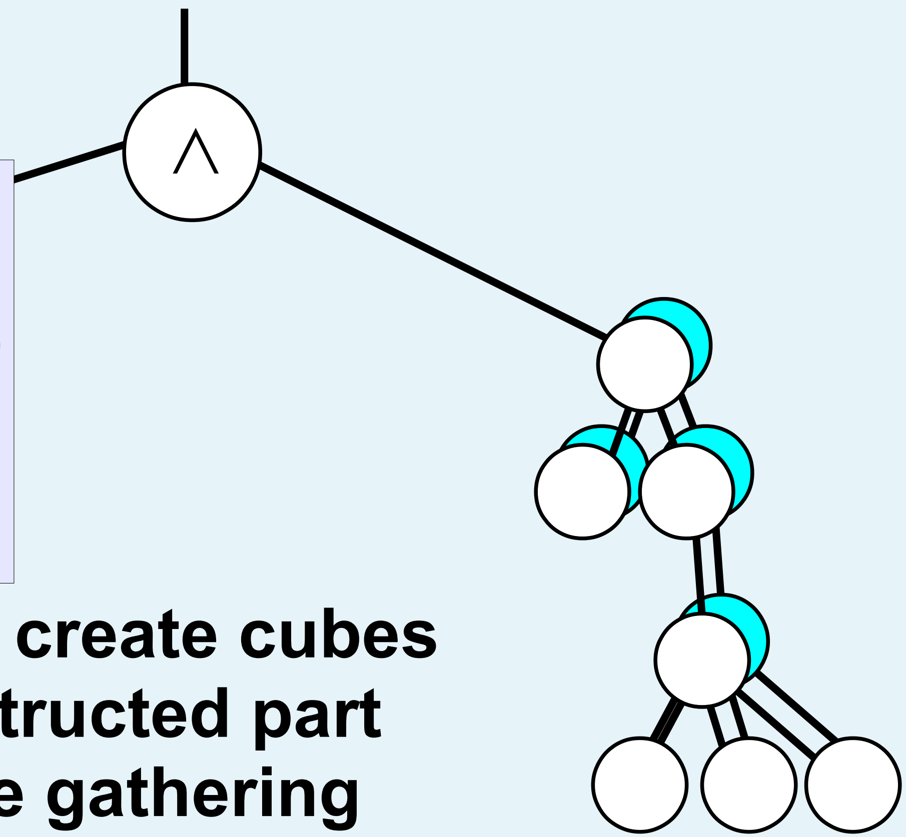
Reconstructing structure

$$
\begin{aligned}
&(a \vee b \vee c)\\
&\wedge(e \vee f)\\
&\wedge(\neg a \vee \neg f \vee c)\\
&\wedge(a \vee d \vee e)\\
&\wedge(\neg a \vee \neg b)\\
&\quad \ldots
\end{aligned}
$$



- **Instead, we could create cubes only for the reconstructed part**
- **Let the usual cube gathering happen in the remainder**

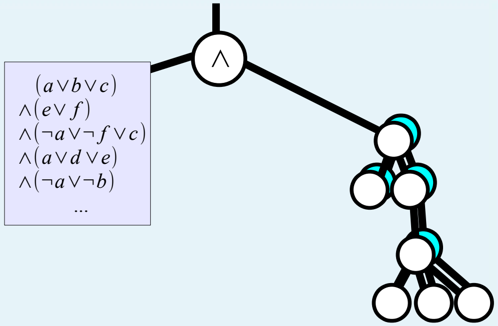
Reconstructing structure



$$
\begin{aligned}
&(a \vee b \vee c) \\
\wedge &(e \vee f) \\
\wedge &(\neg a \vee \neg f \vee c) \\
\wedge &(a \vee d \vee e) \\
\wedge &(\neg a \vee \neg b) \\
&\ldots
\end{aligned}
$$

- **Instead, we could create cubes only for the reconstructed part**
- **Let the usual cube gathering happen in the remainder**

Partial Duality



- Will not gather cubes from definition clauses
- Creates new universal variables to make cubes more expressive
- No efficiency loss on poorly reconstructed instances
- Complete dual propagation on fully reconstructed instances

Plaisted-Greenbaum

- Instead of equivalences, uses implication for variable definitions
- Can be reconstructed using simple syntactic properties

$(\alpha_1 \vee x)$	$(\beta_1 \vee \neg x)$
$(\alpha_2 \vee x)$	$(\beta_2 \vee \neg x)$
$(\alpha_3 \vee x)$	$(\beta_3 \vee \neg x)$
$(\alpha_4 \vee x)$	$(\beta_4 \vee \neg x)$
...	...

$$\begin{array}{ll} (\alpha_1 \vee x) & (\beta_1 \vee \neg x) \\ (\alpha_2 \vee x) & (\beta_2 \vee \neg x) \\ (\alpha_3 \vee x) & (\beta_3 \vee \neg x) \\ (\alpha_4 \vee x) & (\beta_4 \vee \neg x) \\ \ldots & \ldots \end{array}$$

x is tailing

$$\underbrace{\begin{array}{c}(\alpha_1 \vee x)\\(\alpha_2 \vee x)\\(\alpha_3 \vee x)\\(\alpha_4 \vee x)\\ \ldots\end{array}}_{x \text{ is tailing}} \qquad \begin{array}{c}(\beta_1 \vee \neg x)\\(\beta_2 \vee \neg x)\\(\beta_3 \vee \neg x)\\(\beta_4 \vee \neg x)\\ \ldots\end{array}$$

Then: $(\neg\alpha_1 \vee \neg\alpha_2 \vee \neg\alpha_3 \vee \ldots) \rightarrow x$

$$(\alpha_1 \vee x) \qquad (\beta_1 \vee \neg x)$$
$$(\alpha_2 \vee x) \qquad (\beta_2 \vee \neg x)$$
$$(\alpha_3 \vee x) \qquad (\beta_3 \vee \neg x)$$
$$(\alpha_4 \vee x) \qquad (\beta_4 \vee \neg x)$$
$$\underbrace{\quad \ldots \quad}_{x \text{ is tailing}} \qquad \ldots$$

Then: $(\neg\alpha_1 \vee \neg\alpha_2 \vee \neg\alpha_3 \vee \ldots) \rightarrow x$

Set the dual for x to be a new universal u such that:

$$(\alpha_1 \wedge \alpha_2 \wedge \alpha_3 \wedge \ldots) \rightarrow \neg u$$

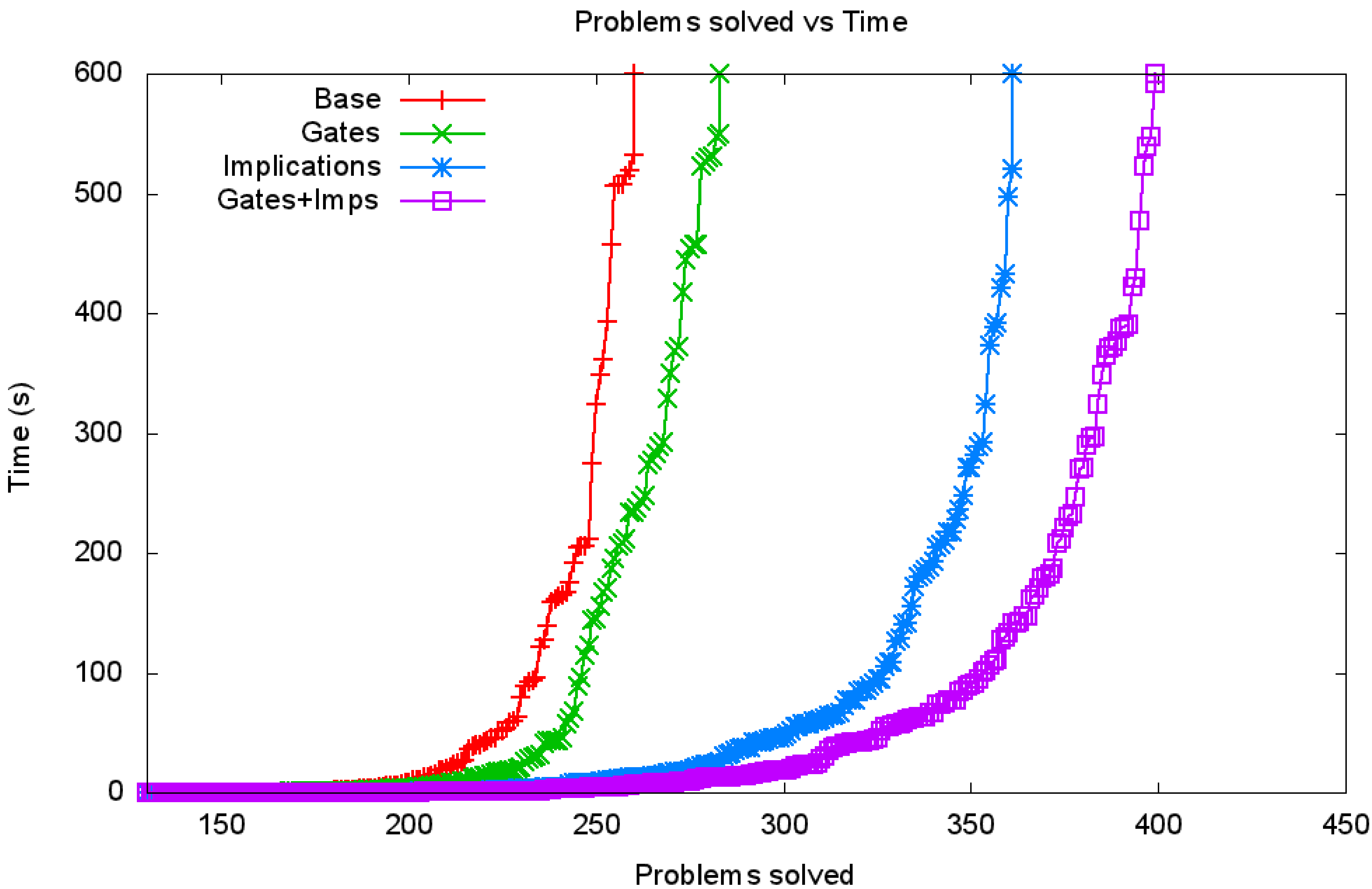
$$\underbrace{\begin{array}{c} (\alpha_1 \vee x) \\ (\alpha_2 \vee x) \\ (\alpha_3 \vee x) \\ (\alpha_4 \vee x) \\ \ldots \end{array}}_{x \text{ is tailing}} \qquad\qquad \begin{array}{c} (\beta_1 \vee \neg x) \\ (\beta_2 \vee \neg x) \\ (\beta_3 \vee \neg x) \\ (\beta_4 \vee \neg x) \\ \ldots \end{array}$$

Then: $(\neg\alpha_1 \vee \neg\alpha_2 \vee \neg\alpha_3 \vee \ldots) \rightarrow x$

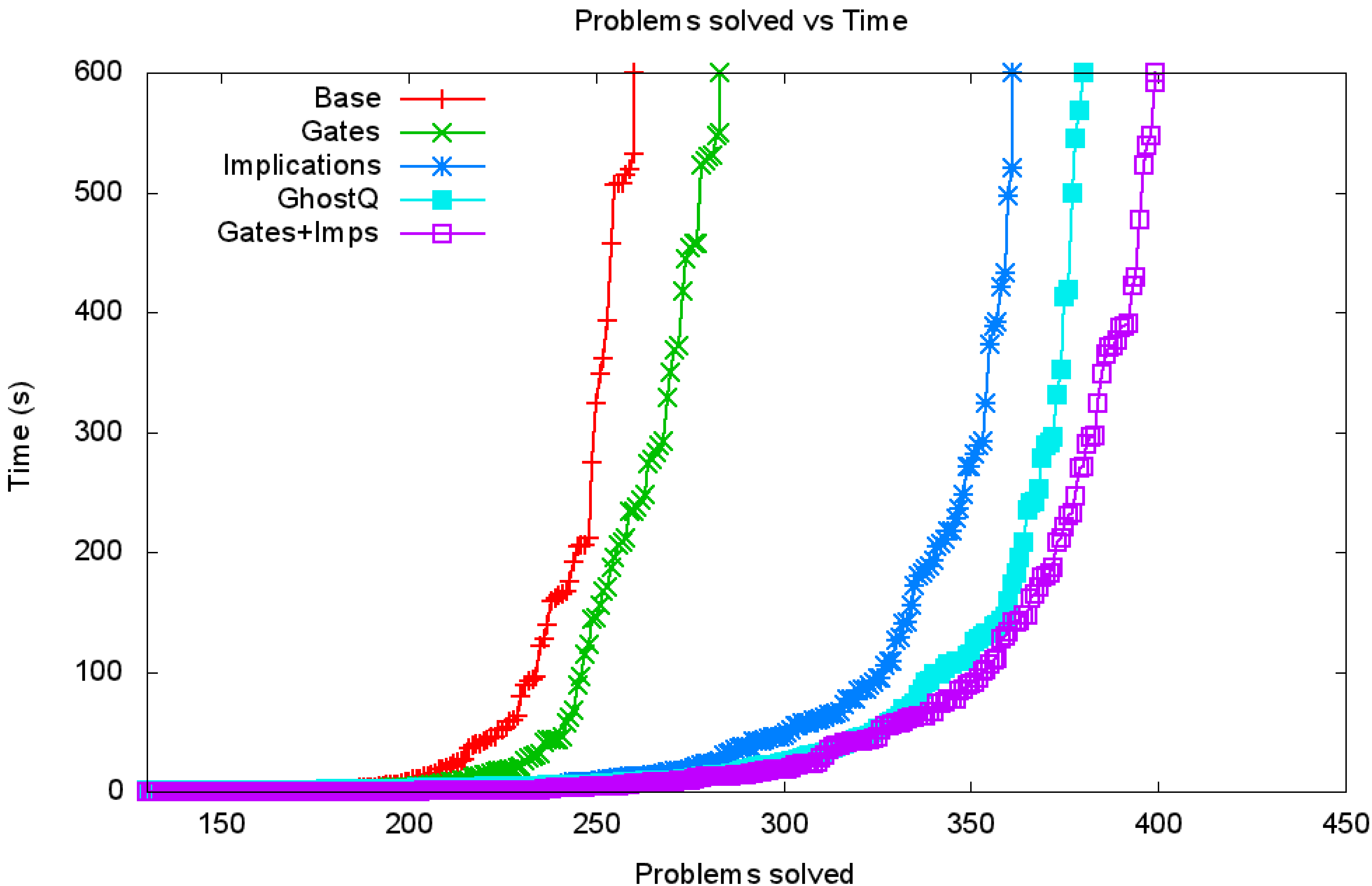
Set the dual for x **to be a new universal** u **such that:**

$$(\alpha_1 \wedge \alpha_2 \wedge \alpha_3 \wedge \ldots) \rightarrow \neg u$$

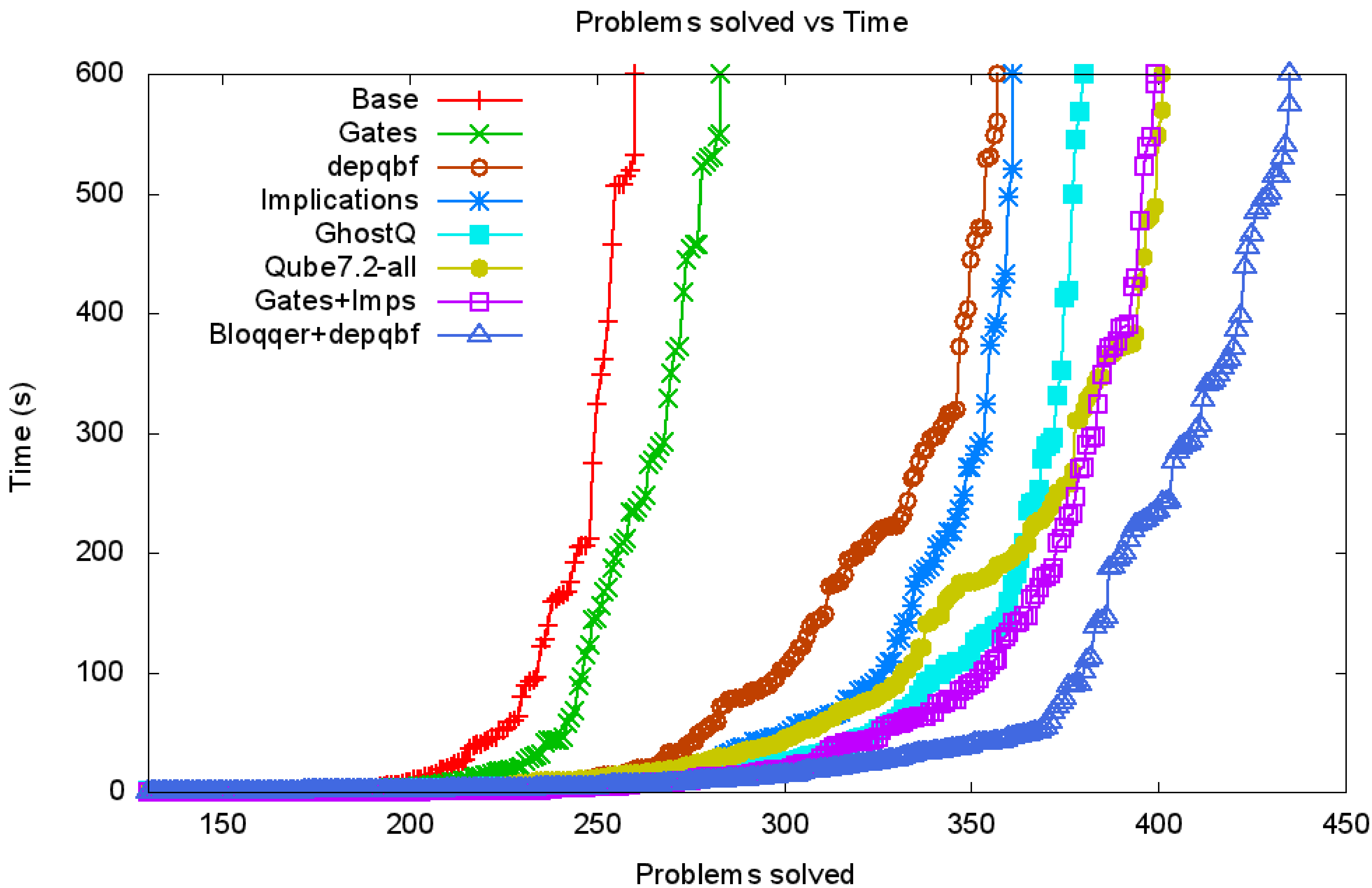
Intuitively: set $(\neg\alpha_1 \vee \neg\alpha_2 \vee \neg\alpha_3 \vee \ldots) \equiv x$
and then remove blocked clauses and cubes



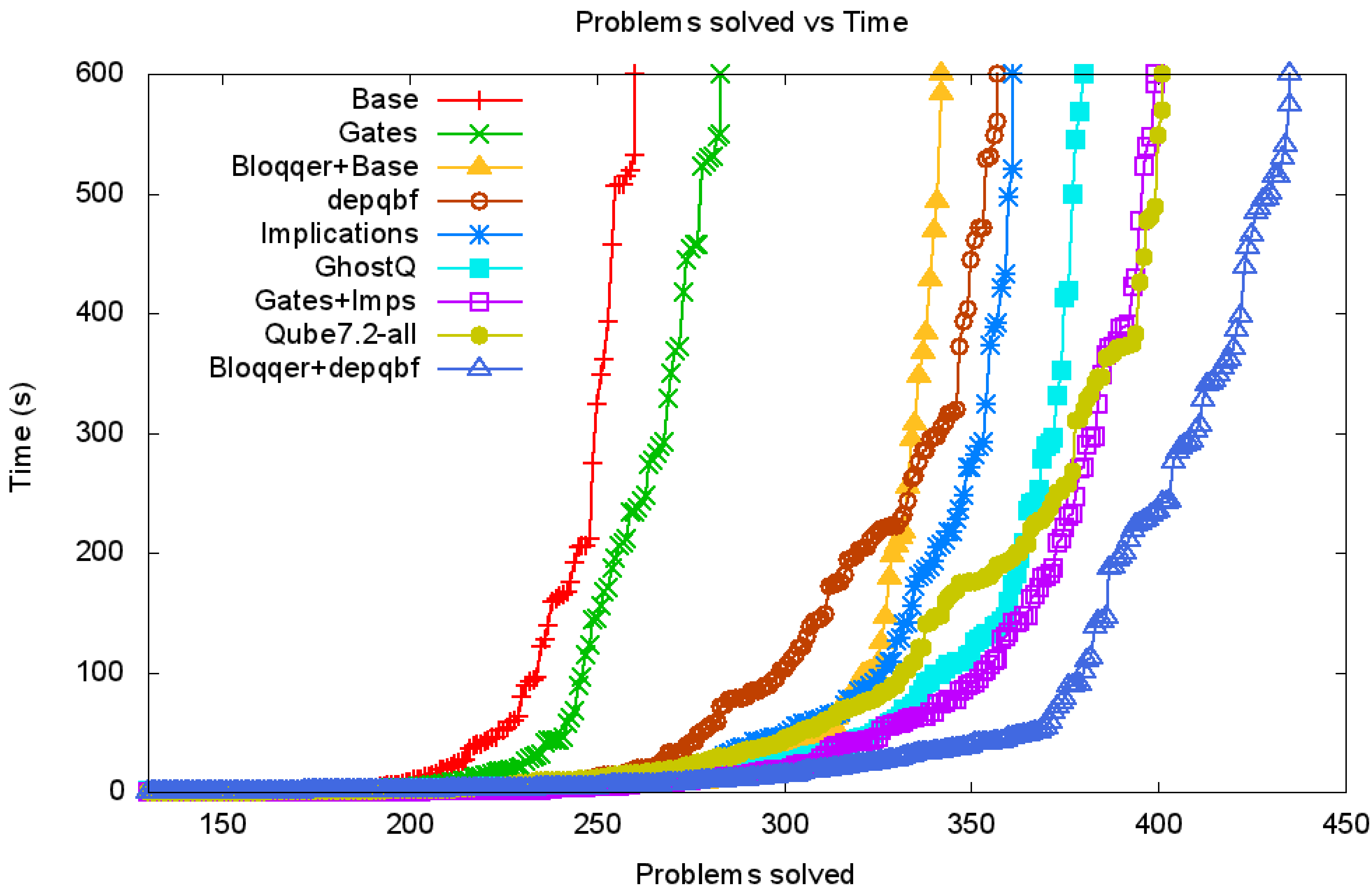
Problems solved vs Time
Base
Gates
Implications
Gates+Imps
Time (s)
0
100
200
300
400
500
600
150
200
250
300
350
400
450
Problems solved



Problems solved vs Time
Base
Gates
Implications
GhostQ
Gates+Imps
Time (s)
Problems solved
0
100
200
300
400
500
600
150
200
250
300
350
400
450



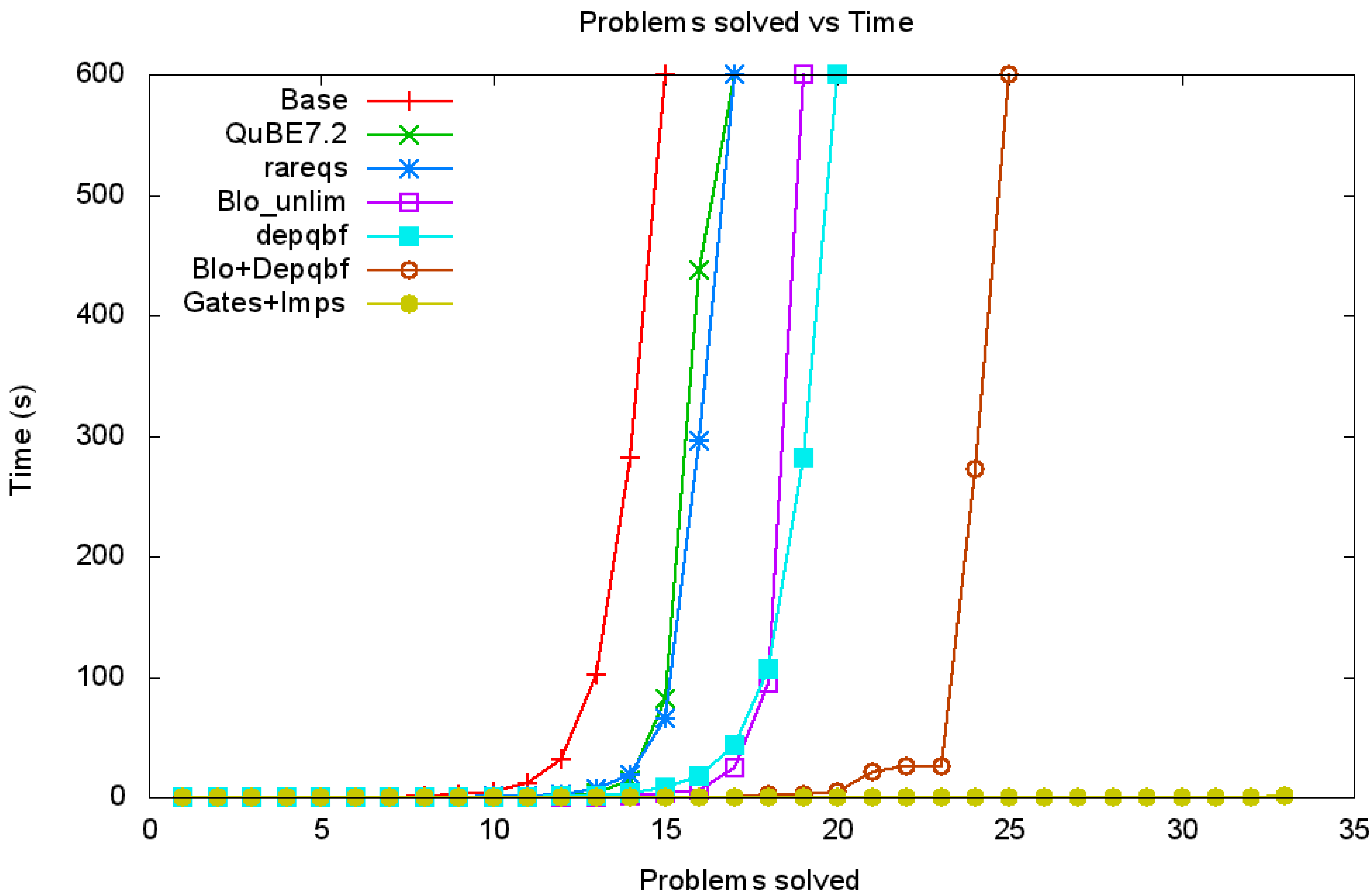
Problems solved vs Time
Time (s)
Problems solved
600
500
400
300
200
100
0
150
200
250
300
350
400
450
Base
Gates
depqbf
Implications
GhostQ
Qube7.2-all
Gates+Imps
Bloqqer+depqbf


Problems solved vs Time
Base
Gates
Bloqqer+Base
depqbf
Implications
GhostQ
Gates+Imps
Qube7.2-all
Bloqqer+depqbf
Time (s)
600
500
400
300
200
100
0
150
200
250
300
350
400
450
Problems solved

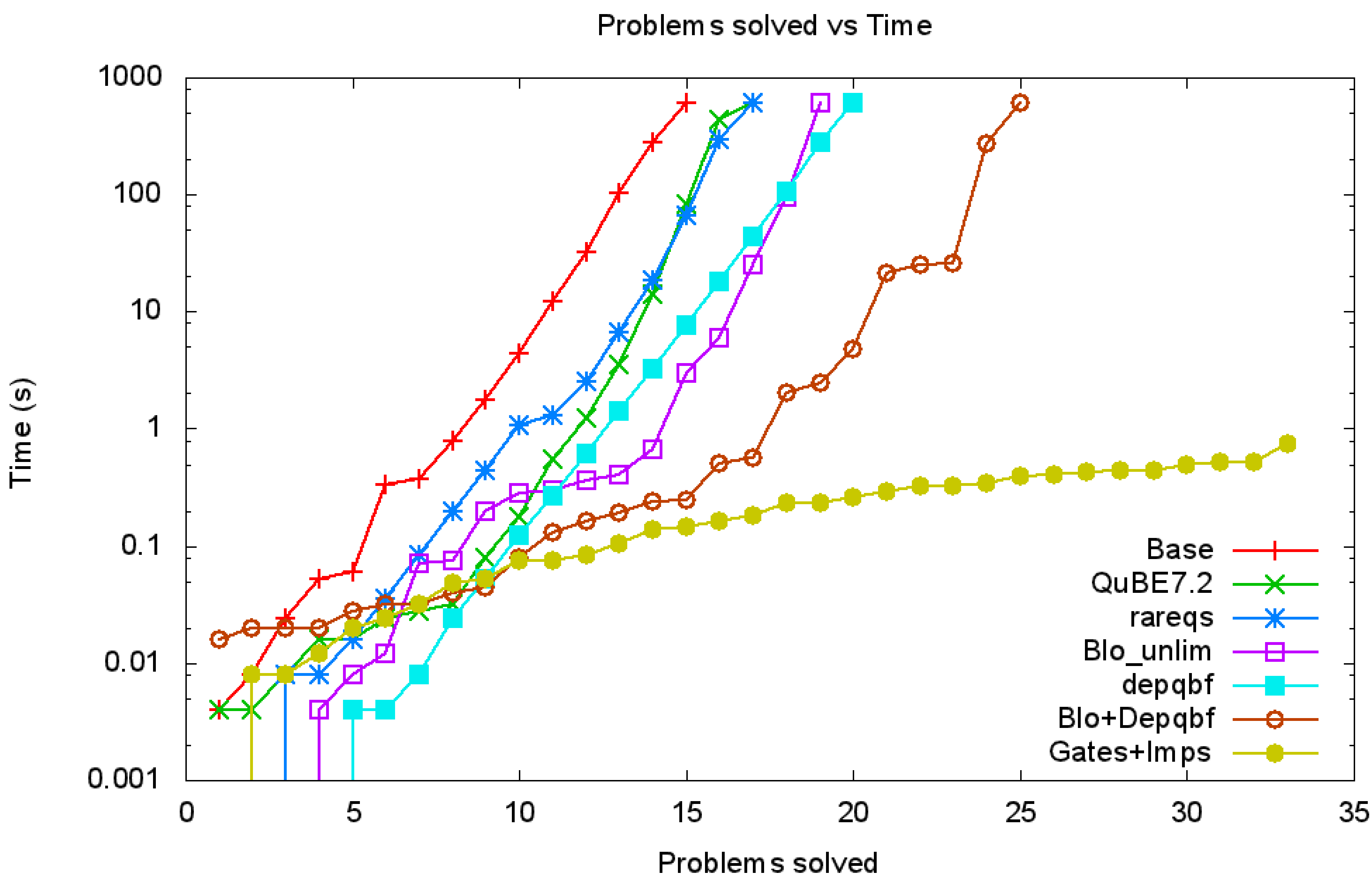
Extreme example

- A family of benchmarks with parameter n

$$
\begin{aligned}
&\exists e \forall x_1 x_2 x_3 \ldots x_n \\
&\qquad\qquad (x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \ldots \oplus x_n) \\
&\qquad\quad \vee (e \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \ldots \oplus x_n)
\end{aligned}
$$



Problems solved vs Time
Time (s)
Problems solved
Base
QuBE7.2
rareqs
Blo_unlim
depqbf
Blo+Depqbf
Gates+Imps
0
100
200
300
400
500
600
0
5
10
15
20
25
30
35



Problems solved vs Time
1000
100
10
1
0.1
0.01
0.001
Time (s)
0
5
10
15
20
25
30
35
Problems solved
Base
QuBE7.2
rareqs
Blo_unlim
depqbf
Blo+Depqbf
Gates+Imps

Conclusions

- CNF does not provide enough information to reason about solutions
- It is possible to use existing incomplete methods to partially reconstruct CNF. That information can be used such that:
  - The better the reconstruction, the more beneficial it is
  - If reconstruction is poor, efficiency is not lost
- Plaisted-Greenbaum encoding can also be reconstructed

Questions?