On Minimal Constraint Networks

Georg Gottlob

Minimal Constraint Networks

Montanari 1974:

To each binary constraint network N, there exists a unique equivalent complete network M(N) such that each pair of values of a constraint in M(N) belongs to some solution.

This ***minimal*** network M(N) can be computed by projecting sol(N) to all pairs of distinct variables.

Example →

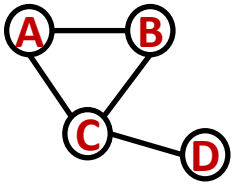
A	B
a1	b1
a1	b2
a1	b3
a2	b1
a3	b4

B	C
b1	c1
b1	c2
b2	c2
b3	c1
b4	c1

A	C
a1	c2
a2	c1
a3	c1
a3	c2

C	D
c1	d2
c2	d1

Binary constraint network N



A	B
a1	b1
a1	b2
a1	b3
a2	b1
a3	b4

⋈

B	C
b1	c1
b1	c2
b2	c2
b3	c1
b4	c1

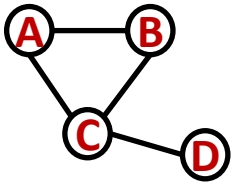
⋈

A	C
a1	c2
a2	c1
a3	c1
a3	c2

⋈

C	D
c1	d2
c2	d1

Binary constraint network N



A	B	C	D
a1	b1	c2	d1
a1	b2	c2	d1
a2	b1	c1	d2
a3	b4	c1	d2

sol(N)

A	B
a1	b1
a1	b2
a2	b1
a3	b4

B	C
b1	c1
b1	c2
b2	c2
b4	c1

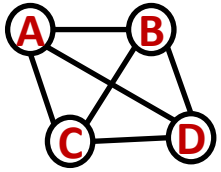
A	C
a1	c2
a2	c1
a3	c1

A	D
a1	d1
a2	d2
a3	d2

B	D
b1	d1
b1	d2
b2	d1
b4	d2

C	D
c1	d2
c2	d1

Minimal network M(N)



A	B
a1	b1
a1	b2
a1	b3
a2	b1
a3	b4

⋈

B	C
b1	c1
b1	c2
b2	c2
b3	c1
b4	c1

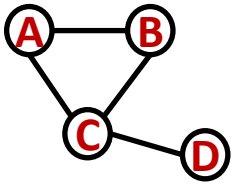
⋈

A	C
a1	c2
a2	c1
a3	c1
a3	c2

⋈

C	D
c1	d2
c2	d1

Binary constraint network N



A	B	C	D
a1	b1	c2	d1
a1	b2	c2	d1
a2	b1	c1	d2
a3	b4	c1	d2

sol(N)

Definition generalizes to k-ary constraint networks…

A	B
a1	b1
a1	b2
a2	b1
a3	b4

B	C
b1	c1
b1	c2
b2	c2
b4	c1

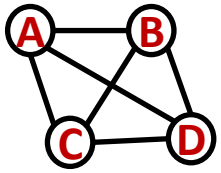
A	C
a1	c2
a2	c1
a3	c1

A	D
a1	d1
a2	d2
a3	d2

B	D
b1	d1
b1	d2
b2	d1
b4	d2

C	D
c1	d2
c2	d1

Minimal network M(N)



Facts about minimal networks

- By definition, sol(M(N)) = sol(N)
- M(N) is empty iff N has no solution.
- Computing M(N) from N is NP-hard [Montanari; Mackworth]
- Recognizing minimality of a network NP-hard [Gaur-95]

More Facts about minimal networks

- Idempotency of M-operator: M(M(N)) = M(N)
- M(N) is the intersection of all networks *equivalent* to M(N) (defined over the same complete schema as M(N))
- M(N) is the intersection of all networks *weaker than* M(N) (defined over the same complete schema as M(N))

Usefulness of minimal networks: Knowledge Compilation

- k-variable queries can be answered in PTIME based on the k-ary minimal network:

  $\exists x \forall y$ (partof(y,x) $\rightarrow$ $\neg$samecolor(x,y))

- Relevant to product configuration problems (e.g. k=3)

  cartype= *suv*, engine=*gas*, make=(*volvo or vw*)

Polynomial space!

Usefulness of minimal networks: Knowledge Compilation

- k-variable queries can be answered in PTIME based on the k-ary minimal network:

  $\exists x \forall y$ (partof(y,x) → ¬samecolor(x,y))

- Relevant to product configuration problems (e.g. k=3)

  cartype= *suv*, engine=*gas*, make=(*volvo or vw*)

Usefulness of minimal networks: Knowledge Compilation

- k-variable queries can be answered in PTIME based on the k-ary minimal network:

  $\exists x \forall y$ (partof(y,x) → ¬samecolor(x,y))

- Relevant to product configuration problems

  cartype= *suv*, engine=*gas*, make=(*volvo or vw*)

Express three wishes, and I will check them in polynomial time!

Main Problem Studied

Given a nonempty minimal constraint network N, compute one solution to N.

How complex is this problem?

70es-90es: Many believed it to be tractable.

Some even believed all solutions to N can be generated by simple backtrack-free search.

But this is not the case.

A	B
a1	b1
a1	b2
a1	b3
a2	b1
a3	b4

⋈

B	C
b1	c1
b1	c2
b2	c2
b3	c1
b4	c1

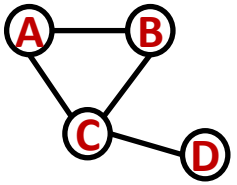
⋈

A	C
a1	c2
a2	c1
a3	c1
a3	c2

⋈

C	D
c1	d2
c2	d1

Binary constraint network N



A	B	C	D
a1	b1	c2	d1
a1	b2	c2	d1
a2	b1	c1	d2
a3	b4	c1	d2

sol(N)

A	B9
a1	b1
a1	b2
a2	b1
a3	b4

B	C
b1	c1
b1	c2
b2	c2
b4	c1

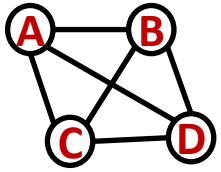
A	C
a1	c2
a2	c1
a3	c1

A	D
a1	d1
a2	d2
a3	d2

B	D
b1	d1
b1	d2
b2	d1
b4	d2

C	D
c1	d2
c2	d1

Minimal network M(N)



A	B
a1	b1
a1	b2
a1	b3
a2	b1
a3	b4

⋈

B	C
b1	c1
b1	c2
b2	c2
b3	c1
b4	c1

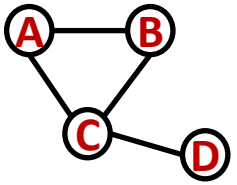
⋈

A	C
a1	c2
a2	c1
a3	c1
a3	c2

⋈

C	D
c1	d2
c2	d1

Binary constraint network N



A	B	C	D
a1	b1	c2	d1
a1	b2	c2	d1
a2	b1	c1	d2
a3	b4	c1	d2

sol(N)

A	B
a1	b1
a1	b2
a2	b1
a3	b4

B	C
b1	c1
b1	c2
b2	c2
b4	c1

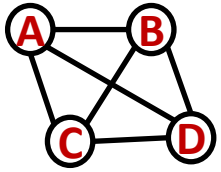
A	C
a1	c2
a2	c1
a3	c1

A	D
a1	d1
a2	d2
a3	d2

B	D
b1	d1
b1	d2
b2	d1
b4	d2

C	D
c1	d2
c2	d1

Minimal network M(N)



A	B
a1	b1
a1	b2
a1	b3
a2	b1
a3	b4

⋈

B	C
b1	c1
b1	c2
b2	c2
b3	c1
b4	c1

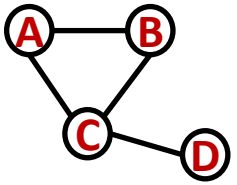
⋈

A	C
a1	c2
a2	c1
a3	c1
a3	c2

⋈

C	D
c1	d2
c2	d1

Binary constraint network N



A	B	C	D
a1	b1	c2	d1
a1	b2	c2	d1
a2	b1	c1	d2
a3	b4	c1	d2

sol(N)

A	B
a1	b1
a1	b2
a2	b1
a3	b4

B	C
b1	c1
b1	c2
b2	c2
b4	c1

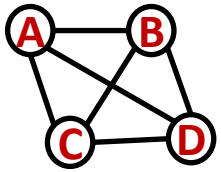
A	C
a1	c2
a2	c1
a3	c1

A	D
a1	d1
a2	d2
a3	d2

B	D
b1	d1
b1	d2
b2	d1
b4	d2

C	D
c1	d2
c2	d1

Minimal network M(N)



A	B
a1	b1
a1	b2
a1	b3
a2	b1
a3	b4

⋈

B	C
b1	c1
b1	c2
b2	c2
b3	c1
b4	c1

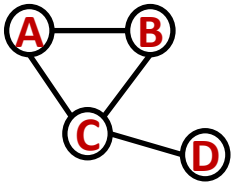
⋈

A	C
a1	c2
a2	c1
a3	c1
a3	c2

⋈

C	D
c1	d2
c2	d1

Binary constraint network N



A	B	C	D
a1	b1	c2	d1
a1	b2	c2	d1
a2	b1	c1	d2
a3	b4	c1	d2

sol(N)

A	B
a1	b1
a1	b2
a2	b1
a3	b4

B	C
b1	c1
b1	c2
b2	c2
b4	c1

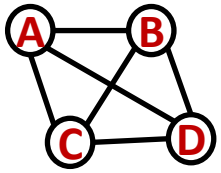
A	C
a1	c2
a2	c1
a3	c1

A	D
a1	d1
a2	d2
a3	d2

B	D
b1	d1
b1	d2
b2	d1
b4	d2

C	D
c1	d2
c2	d1

Minimal network M(N)



Computing a solution

Are there more sophisticated efficient algorithms ?

Conjecture [Gaur 95]: Tractable

Conjecture [Dechter 03]: Intractable

The problem has remained open until recently.

Progress made by Bessiere (CP Handbk) using [Cros03]:

Theorem [Bessiere 06]: If solutions to minimal networks can be computed via polytime backtrack-free fair strategies, then the Polynomial Hierarchy collapses.

Main Result

Theorem: Computing a solution to a non-empty minimal network is NP-hard.

Caveat: Promise-problem.
Still, NP-hardness is well-defined.

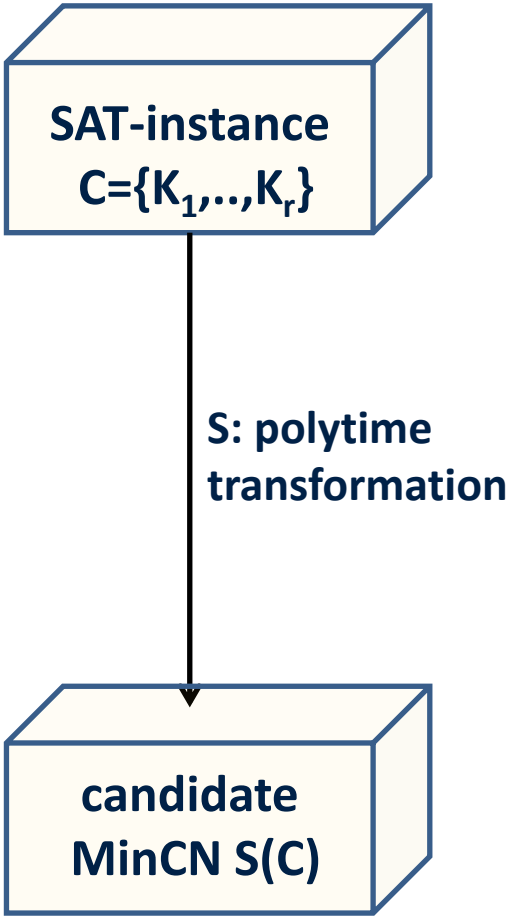
NP-hardness: If there exists a PTIME algorithm ***findsol***, that solves the problem, then NP=P.

Proof Sketch

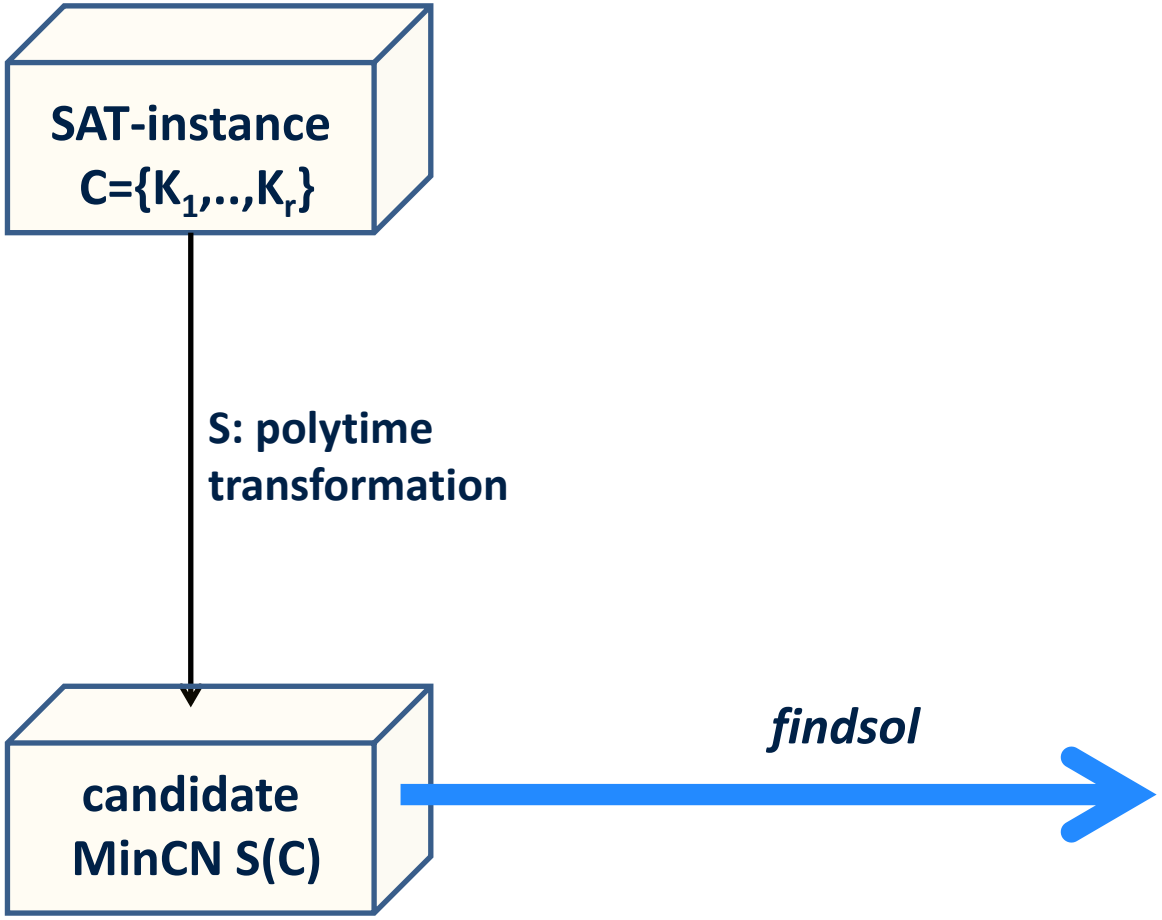
- Show that each SAT instance C can be transformed in polynomial time into an "equivalent" constraint network S(C), such that S(C) is minimal in case C is satisfiable.

- Assume there exists an algorithm ***findsol*** that finds solution to minimal networks in polynomial time. Apply it to S(C) to obtain a PTIME SAT-solver.

Proof Sketch



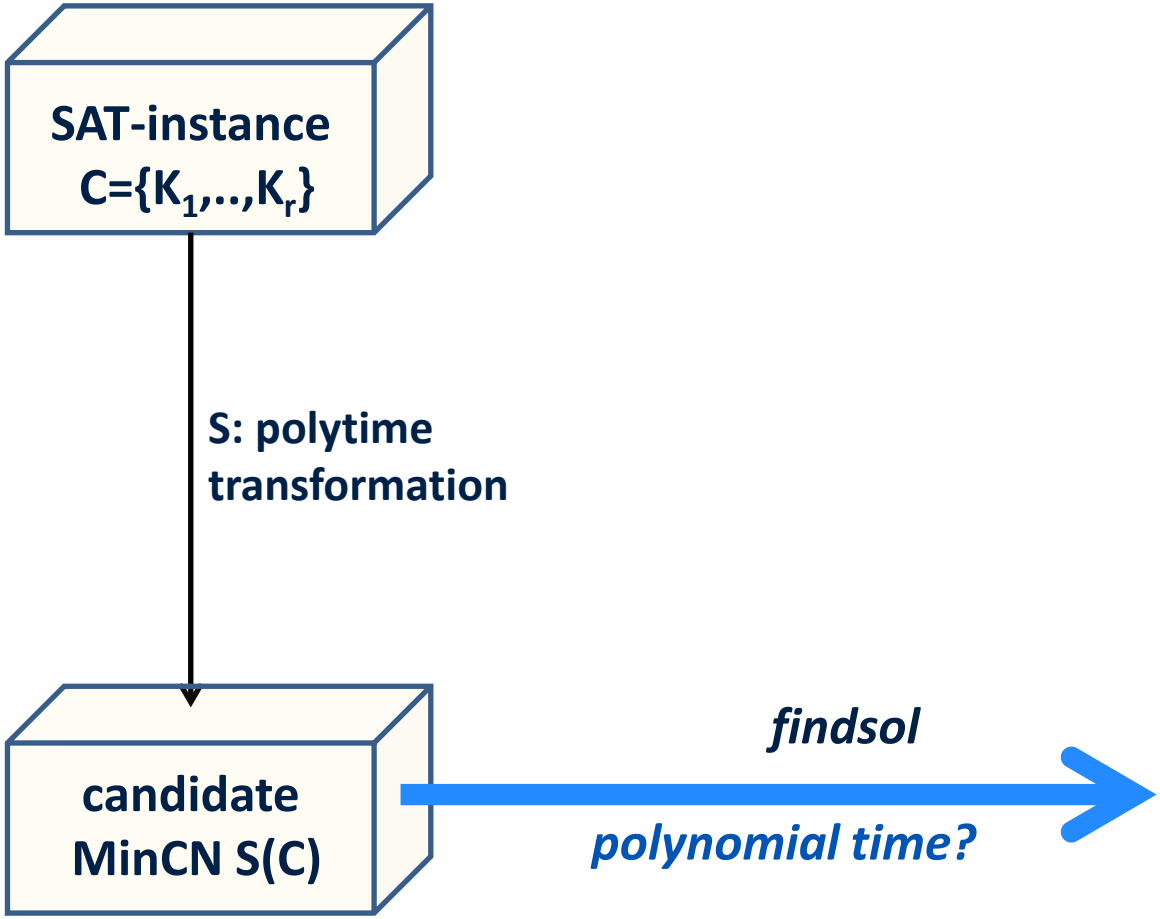
C satisfiable ⇒ S(C) minimal & sol(S(C)) = satisfying truth value assignments of C
C unsatisfiable ⇒ S(C) arbitrary



C satisfiable ⇒ S(C) minimal
C unsatisfiable ⇒ S(C) arbitrary

findsol

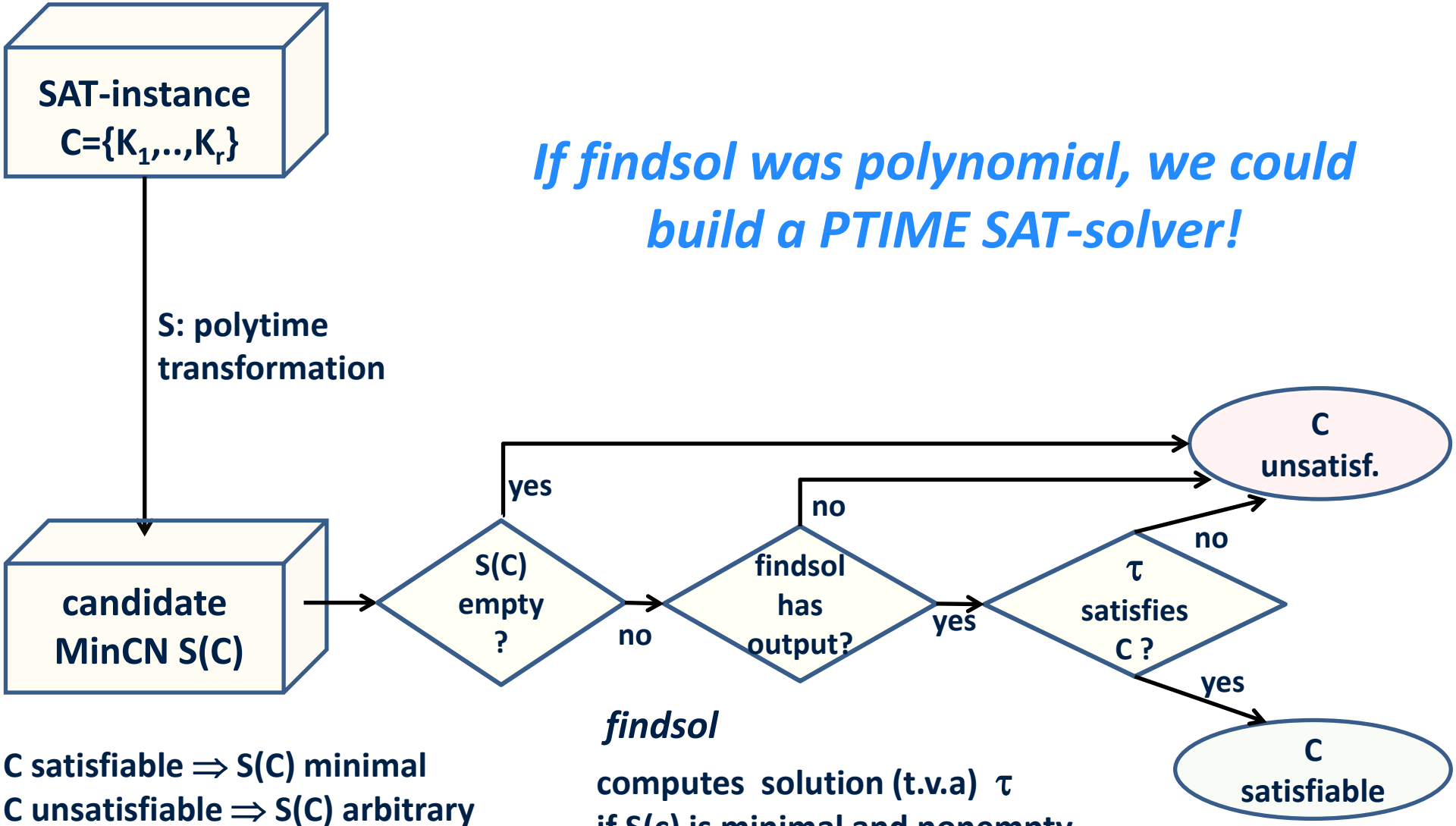
computes solution (t.v.a) τ
if S(c) is minimal and nonempty



C satisfiable ⇒ S(C) minimal
C unsatisfiable ⇒ S(C) arbitrary

findsol

computes solution (t.v.a) τ
if S(c) is minimal and nonempty

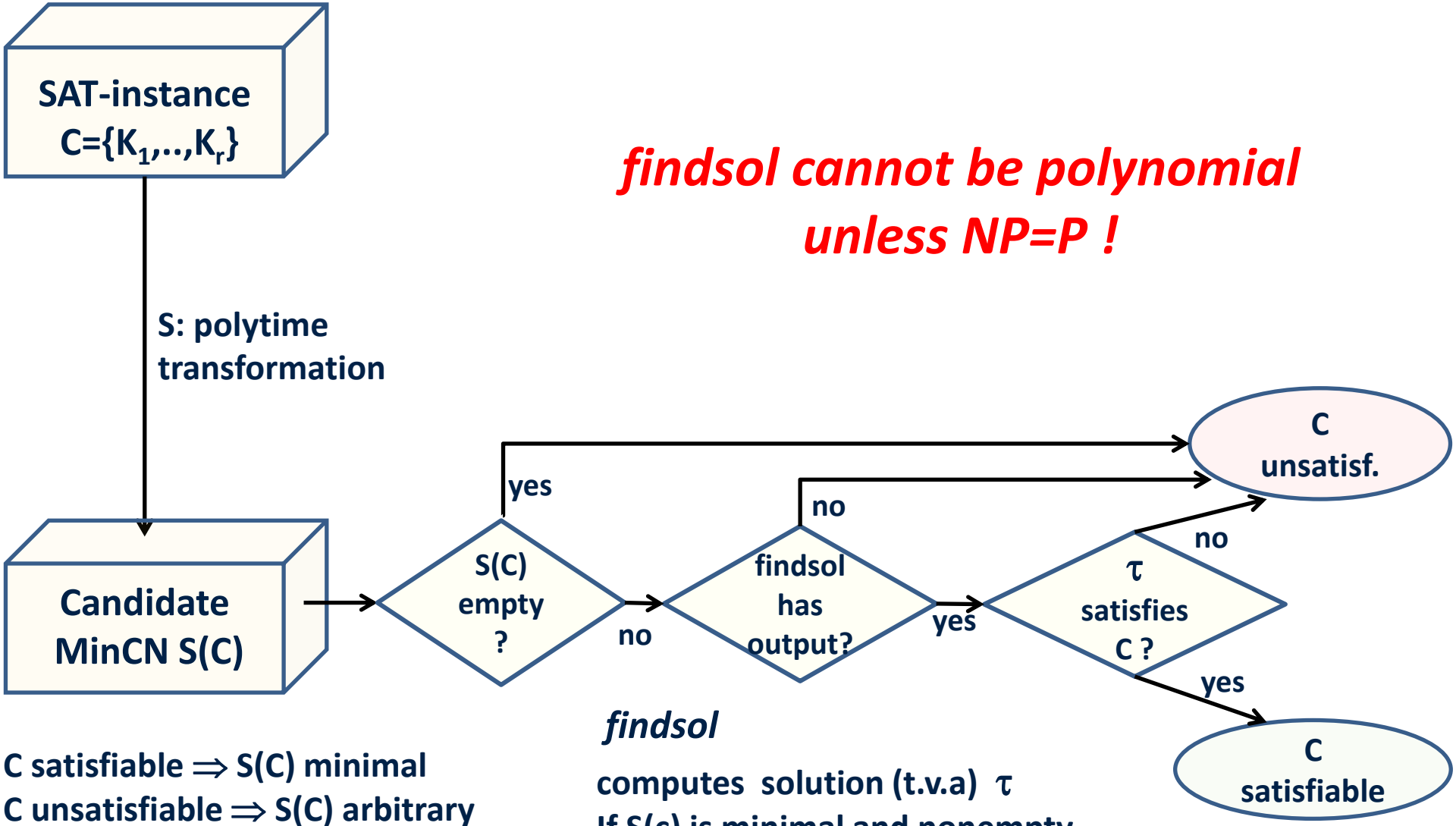


If findsol was polynomial, we could build a PTIME SAT-solver!

C satisfiable ⇒ S(C) minimal
C unsatisfiable ⇒ S(C) arbitrary

findsol

computes solution (t.v.a) τ
if S(c) is minimal and nonempty



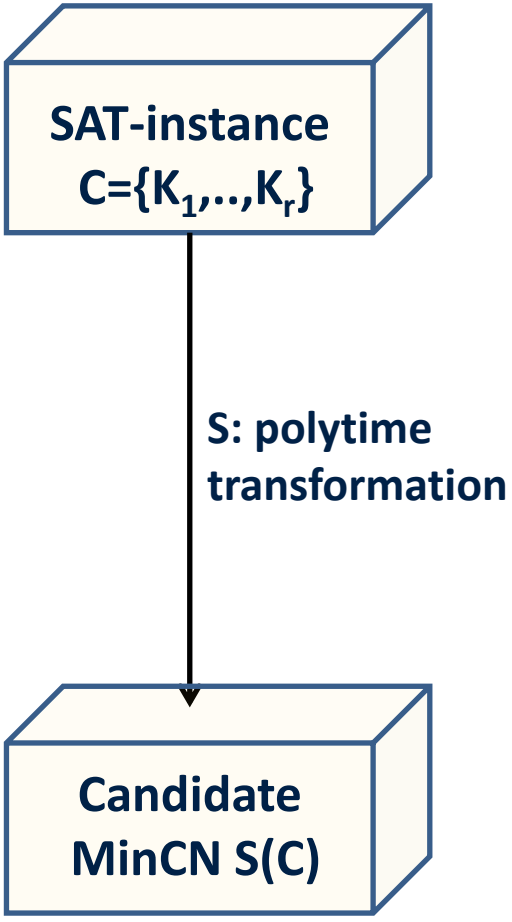
findsol cannot be polynomial unless NP=P !

C satisfiable ⇒ S(C) minimal
C unsatisfiable ⇒ S(C) arbitrary

findsol

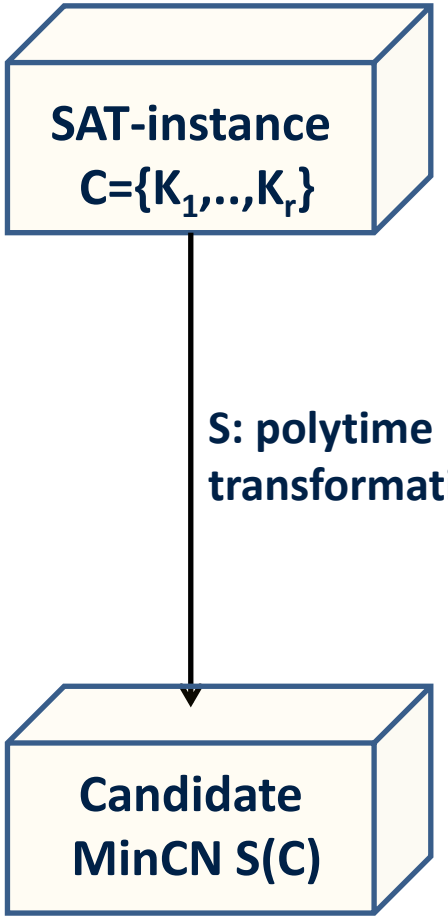
computes solution (t.v.a) τ
If S(c) is minimal and nonempty

Translating C to S(C)



C satisfiable ⇒ S(C) minimal & sol(S(C)) = satisfying truth value assignments of C
C unsatisfiable ⇒ S(C) arbitrary

Translation: First Attempt



$K_1 = p \vee \neg q \vee r$
$K_2 = \neg p \vee \neg q$
$K_3 = q$

Each clause K_i becomes a variable

$Dom(K_1) = \{p, \neg q, r\}$
$Dom(K_2) = \{p, \neg q\}$
$Dom(K_3) = \{q\}$

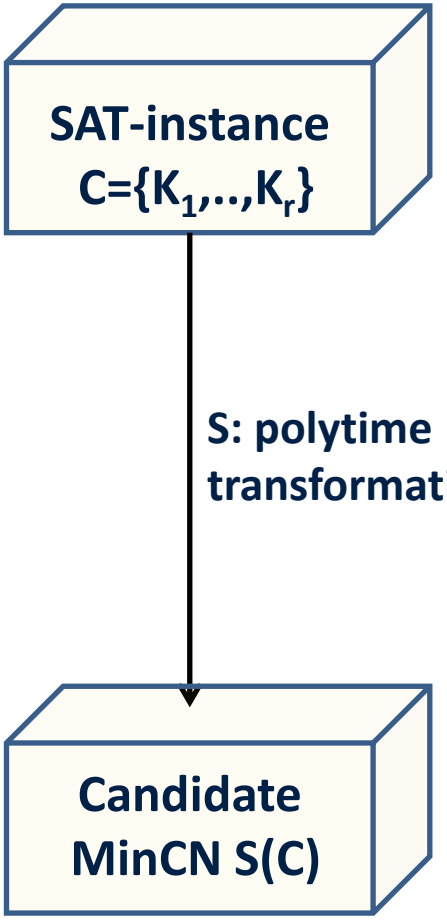
K_1	K_2
p	$\neg q$
$\neg q$	$\neg p$
$\neg q$	$\neg q$
r	$\neg p$
r	$\neg q$

K_1	K_3
p	q
r	q

K_2	K_3
$\neg p$	q

C satisfiable ⇒ S(C) minimal
C unsatisfiable ⇒ S(C) arbitrary

Translation: First Attempt



$K_1 = p \vee \neg q \vee r$
$K_2 = \neg p \vee \neg q$
$K_3 = q$

K_1	K_2
p	$\neg q$
$\neg q$	$\neg p$
$\neg q$	$\neg q$
r	$\neg p$
r	$\neg q$

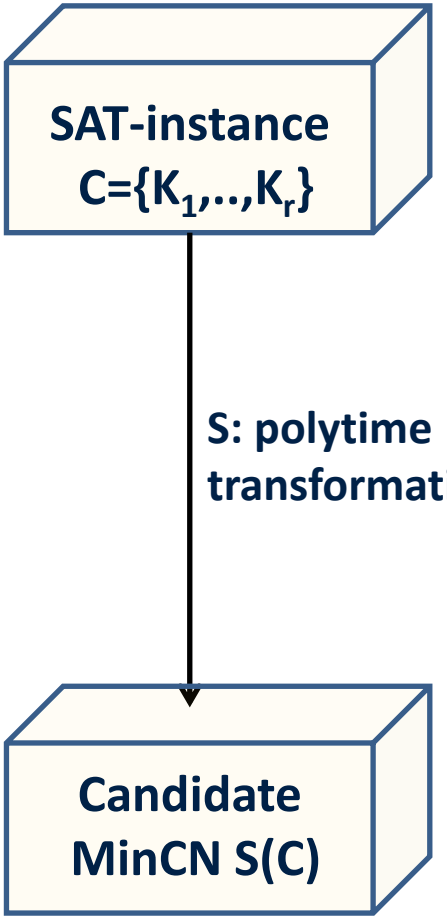
K_1	K_3
p	q
r	q

K_2	K_3
$\neg p$	q

minimal?

C satisfiable ⇒ S(C) minimal
C unsatisfiable ⇒ S(C) arbitrary

Translation: First Attempt



$K_1 = p \vee \neg q \vee r$
$K_2 = \neg p \vee \neg q$
$K_3 = q$

K_1	K_2
~~p~~	~~$\neg q$~~
~~$\neg q$~~	~~$\neg p$~~
~~$\neg q$~~	~~$\neg q$~~
r	$\neg p$
~~r~~	~~$\neg q$~~

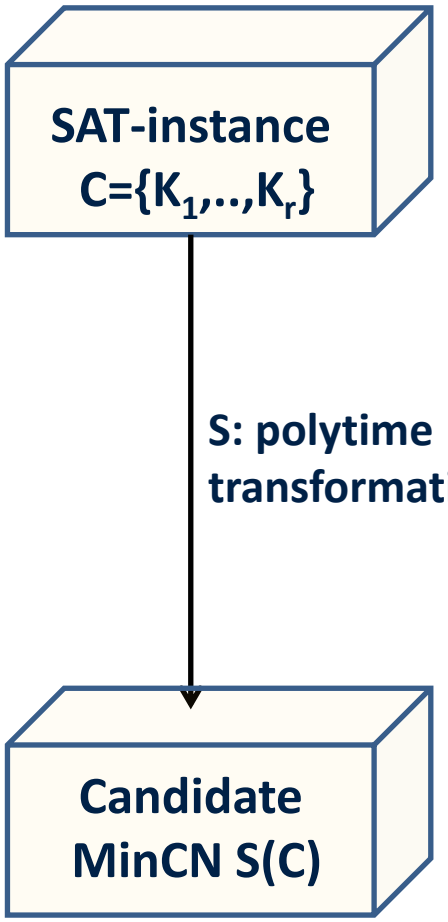
K_1	K_3
~~p~~	~~q~~
r	q

K_2	K_3
$\neg p$	q

Not minimal!
Eliminating useless tuples
Is NP-hard in general!

C satisfiable ⇒ S(C) minimal
C unsatisfiable ⇒ S(C) arbitrary

Translation: A Better Idea



C satisfiable ⇒ S(C) minimal
C unsatisfiable ⇒ S(C) arbitrary

$K_1 = p \vee \neg q \vee r \vee s$
$K_2 = \neg p \vee \neg q \vee r$
$K_3 = p \vee q \vee \neg r \vee s$

supersymmetric
C

$r_{ij} = (Dom(K_i) \times Dom(K_j))$ – opposite pairs

K_1	K_2
p	¬q
p	r
¬q	¬p
¬q	¬q
¬q	r
r	¬p
r	¬q
r	r
.....	

K_1	K_3
p	p
p	q
p	¬r
¬q	p
¬q	¬r
r	p
r	q
.....	

K_2	K_3
¬p	q
¬p	¬r
¬q	p
¬q	¬r
r	p
r	q
.....	

M(C)

Supersymmetry

Definition: A SAT instance C is ***k-supersymmetric*** if C is either unsatisfiable, or if for each k-tuple of propositional variables $p_1 \ldots p_k$ occurring in C, each arbitrary truth value assignment to $p_1 \ldots p_k$ can be extended to a satisfying truth value assignment to C.

supersymmetric = 2-supersymmetric

Symmetry Lemma

Lemma: For each fixed $k \geq 2$ there is a PTIME reduction that transforms each 3SAT instance C into a k-supersymmetric SAT instance C* such that

C* is satisfiable iff C is satisfiable.

Note: A supersymmetric SAT instance C* can be transformed efficiently into an "equivalent" minimal constraint network S(C*)

The main Theorem follows.

Symmetry Lemma: Proof Idea (k=2)

C

$K_1 = p \vee \neg q \vee r$
$K_2 = \neg p \vee \neg q$
$K_3 = q$

For each atom, we introduce 5 new atoms

$p \rightarrow p_1 \ldots p_5$ $q \rightarrow q_1 \ldots q_5$ $r \rightarrow r_1 \ldots r_5$

Replace each positive atom by 3 corresponding new positive atoms

Replace each negative atom by 3 corresponding new negative atoms

in all possible ways

Symmetry Lemma: Proof Idea (k=2)

C

$K_1 = \quad p \vee \neg q \vee r$
$K_2 = \neg p \vee \neg q$
$K_3 = \quad q$

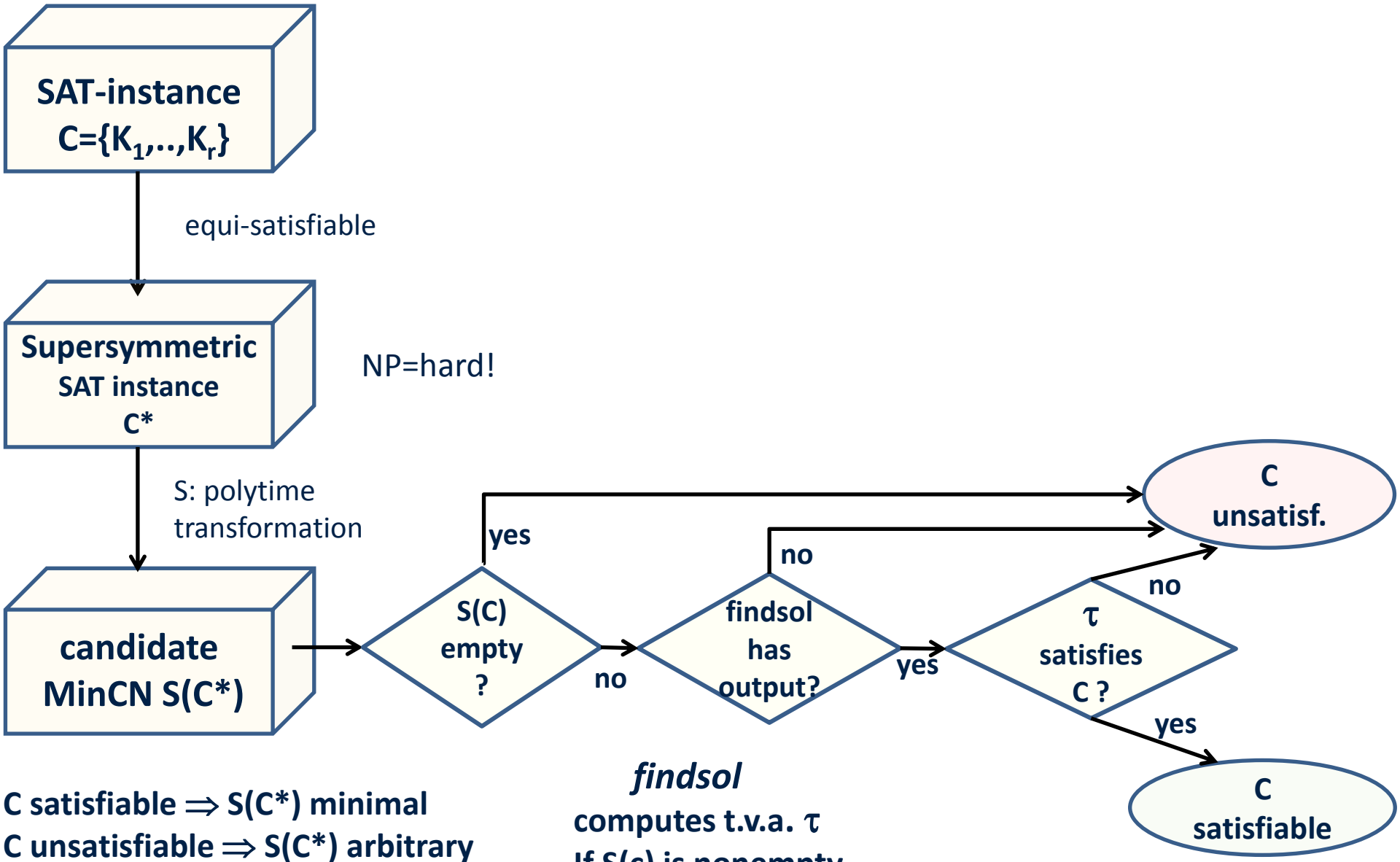
C*

$K_{1,1} = p1 \vee p2 \vee p3 \vee \neg q1 \vee \neg q2 \vee \neg q3 \vee r1 \vee r2 \vee r3$
$K_{1,2} = p1 \vee p2 \vee p3 \vee \neg q1 \vee \neg q2 \vee \neg q3 \vee r1 \vee r2 \vee r4$
$K_{1,3} = p1 \vee p2 \vee p3 \vee \neg q1 \vee \neg q2 \vee \neg q3 \vee r1 \vee r2 \vee r5$
… … … …
$K_{1,55} = p3 \vee p4 \vee p5 \vee \neg q2 \vee \neg q4 \vee \neg q5 \vee r1 \vee r2 \vee r3$
… … … …
… … … …

C* supersymmetric, and C satisfiable iff C* satisfiable.

$\tau(p)$=true $\Leftrightarrow$ τ^* is true for ≥ 3 variables from $p_1 \ldots p_5$
$\tau(p)$=false $\Leftrightarrow$ τ^* is true for < 3 variables from $p_1 \ldots p_5$

Putting it all together



C satisfiable ⇒ S(C*) minimal
C unsatisfiable ⇒ S(C*) arbitrary

findsol
computes t.v.a. τ
If S(c) is nonempty

Further Result

Definition If R is a relation, then $\Pi_2(R)$ denotes the projection of R onto all pairs of R-attributes

R:

A	B	C	D
a1	b1	c2	d1
a1	b2	c2	d1
a2	b1	c1	d2
a3	b4	c1	d2

$\Pi_2(R)$:

A	B
a1	b1
a1	b2
a2	b1
a3	b4

B	C
b1	c1
b1	c2
b2	c2
b4	c1

A	C
a1	c2
a2	c1
a3	c1

A	D
a1	d1
a2	d2
a3	d2

B	D
b1	d1
b1	d2
b2	d1
b4	d2

C	D
c1	d2
c2	d1

Conjecture [Dechter and Pearl 92]: Given a relation R, it is NP-hard to decide whether $sol(\Pi_2(R))=R$, i.e., if R is 2-representable.

Theorem: This problem is co-NP-complete.

Note: Closely related to join-dependency checking for database instances.