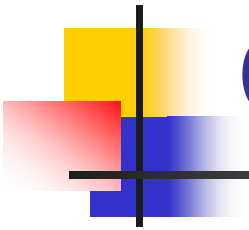
Simultaneous SAT Based Model Checking of Safety Properties

Zurab Khasidashvili, Alexander Nadel, Amit Palti, Ziyad Hanna

Intel, Haifa



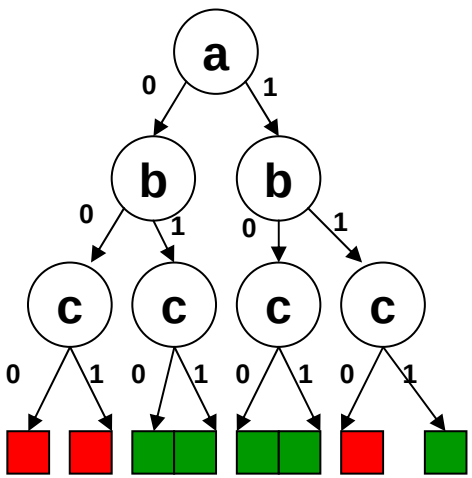
Overview

- Simultaneous propositional satisfiability
  - A DPLL style algorithm for multiple objectives
  - Comparison to incremental SAT
- Simultaneous model checking of safety (invariant) properties
  - New BMC and induction schemes
  - Comparison to previous work
- Experimental results
- Conclusions

Introduction to SAT

- A modern SAT solver invokes backtrack search DPLL algorithm, to find a satisfying assignment to a given CNF formula, or conclude that no such assignment exists

A Search Tree



A CNF Formula

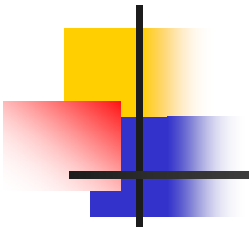
$F = (a + c)(b + c)(\neg a + \neg b + \neg c)$

A clause

A negative literal

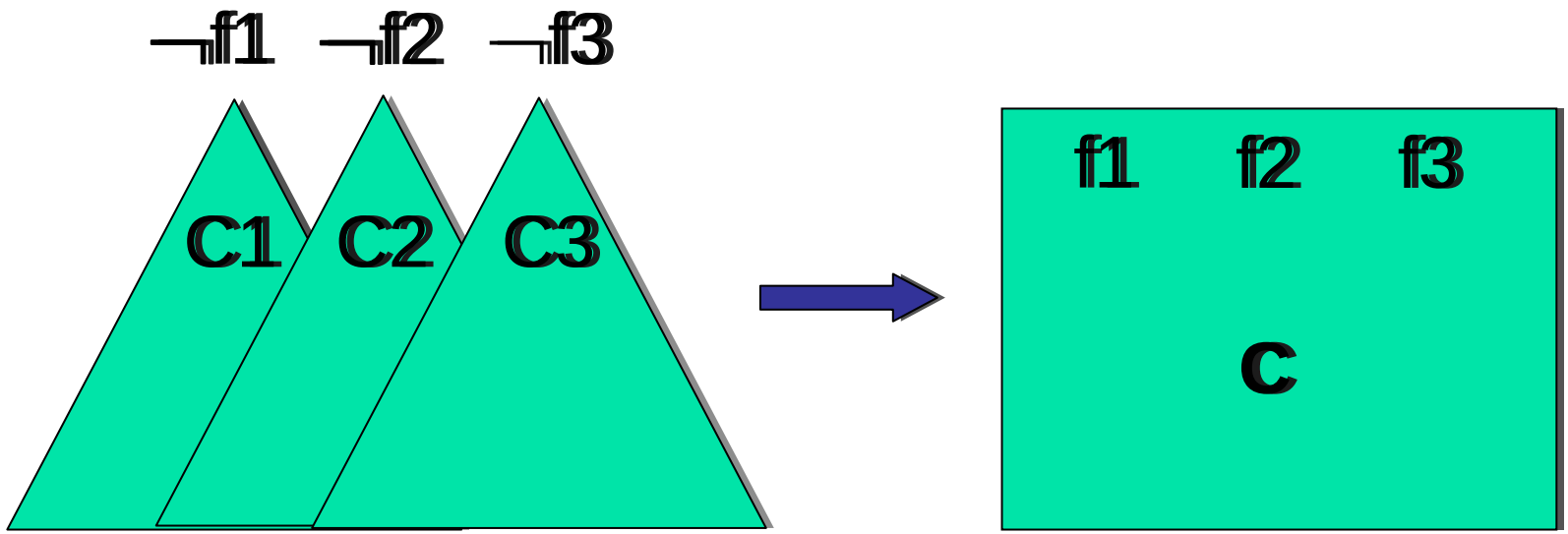
A positive literal

Introduction to SAT Solvers



- Nowadays, DPLL is enhanced by:
  - Boolean Constraints Propagation (BCP)
    - Make forced assignments in unit clauses
  - Conflict-Driven Learning
    - A conflict occurs when a variable must be assigned both 1 and 0 during BCP
    - On conflict, update the formula with a new *conflict clause* (a logical consequence of original clauses)
      - The conflict clause guarantees that the conflict will not reappear during the subsequent search
      - Crucial to prune the search tree
  - Restarts (that help to re-direct the search)

Simultaneous SAT algorithm vs multiple SAT invocations



One aims dis/proving
f1, f2, f3

We aim dis/proving
C→f1, C→f2, C→f3

Simultaneous SAT algorithm: the input & output

- Several formulas to prove – f1,..., fn, called proof objectives or POs;
- All of the POs share the same CNF instance C
  - f1,...,fn are considered as variables; their defining clauses are all included in the CNF C
- The SSAT algorithm takes as input a list f1,...,fn of POs and a CNF instance C, and decides for each i whether fi is a consequence of C
  - If there is an assignment that satisfies C in which fi is assigned false, then fi is falsifiable
  - else fi is valid (is a consequence of C)

SSAT pseudo algorithm

PO1 PO2 PO3 PO4

Unknown

CWPO

Falsified

Valid

SSAT pseudo algorithm (cont)

PO1 PO2 PO3 PO4

Unknown

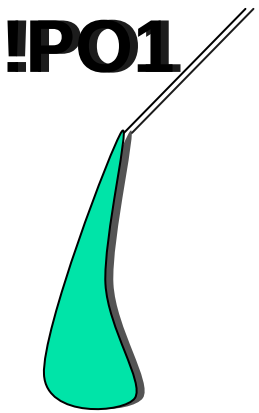
CWPO

Falsified

Valid

SSAT pseudo algorithm (cont)

PO1 PO2 PO3 PO4



Unit clause PO4

Unknown

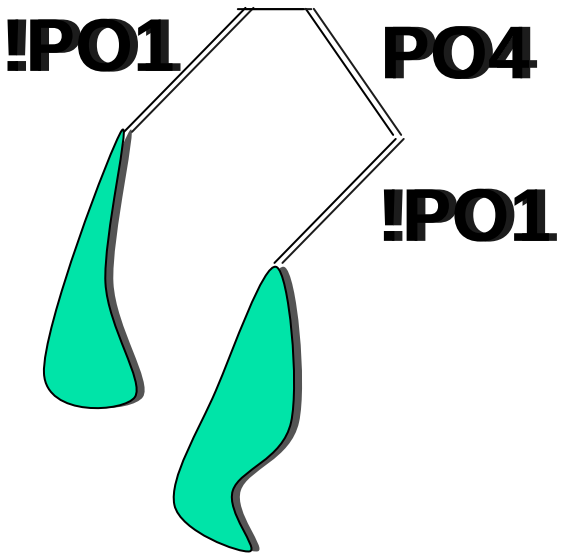
CWPO

Falsified

Valid

SSAT pseudo algorithm (cont)

PO1 PO2 PO3 PO4



Unknown

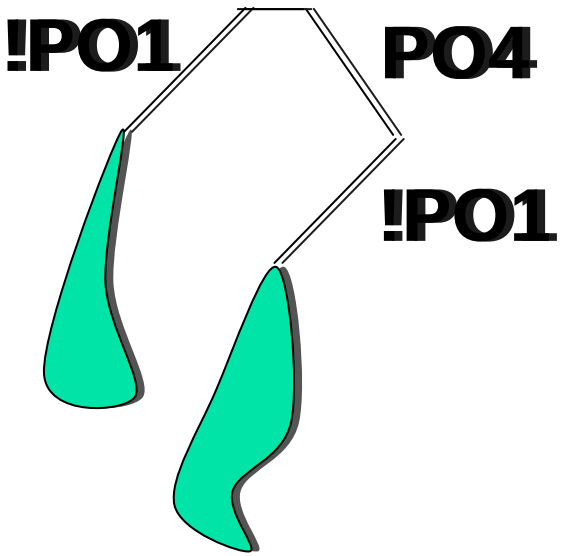
CWPO

Falsified

Valid

SSAT pseudo algorithm (cont)

PO1 PO2 PO3 PO4



Unknown

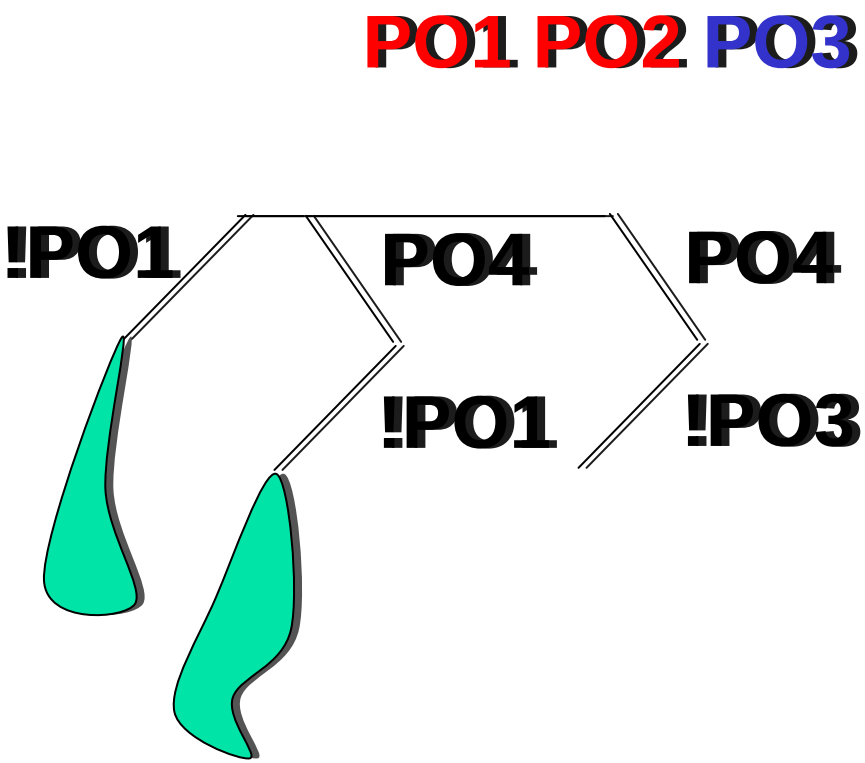
CWPO

Falsified

Valid

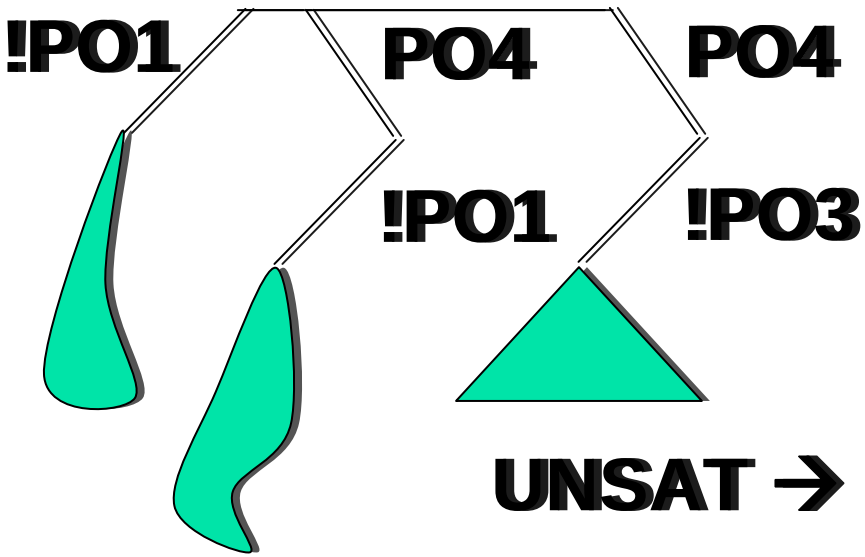
Model: {!PO1, !PO2, PO3, PO4, ...}

SSAT pseudo algorithm (cont)



SSAT pseudo algorithm (cont)

PO1 PO2 PO3 PO4



Unknown

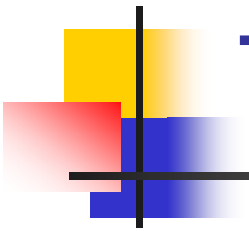
CWPO

Falsified

Valid

UNSAT → PO3 is valid

SSAT important features: The “all watched” principle



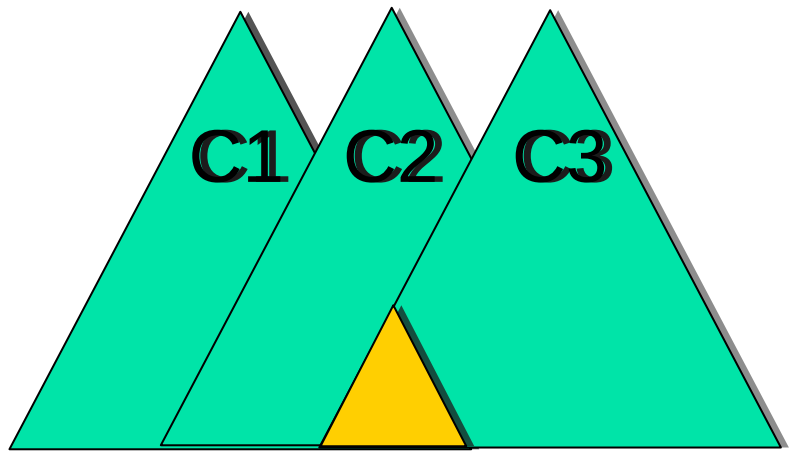
- When the search is oriented to resolve a currently watched PO, we may falsify or prove other POs as well.

SSAT important features: The “one traversal” principle

- In SSAT, the search is organized so that in one (partial) traversal we resolve all the POs -- we never revisit the same sub-space again
  - For example, we will never rediscover the same model (SAT assignment) of CNF C
    - This is guaranteed by the fact that CWPO was true in all previous models
  - The conflict clauses prevent the search from re-entering the explored space again
  - It is safe however to use re-starts

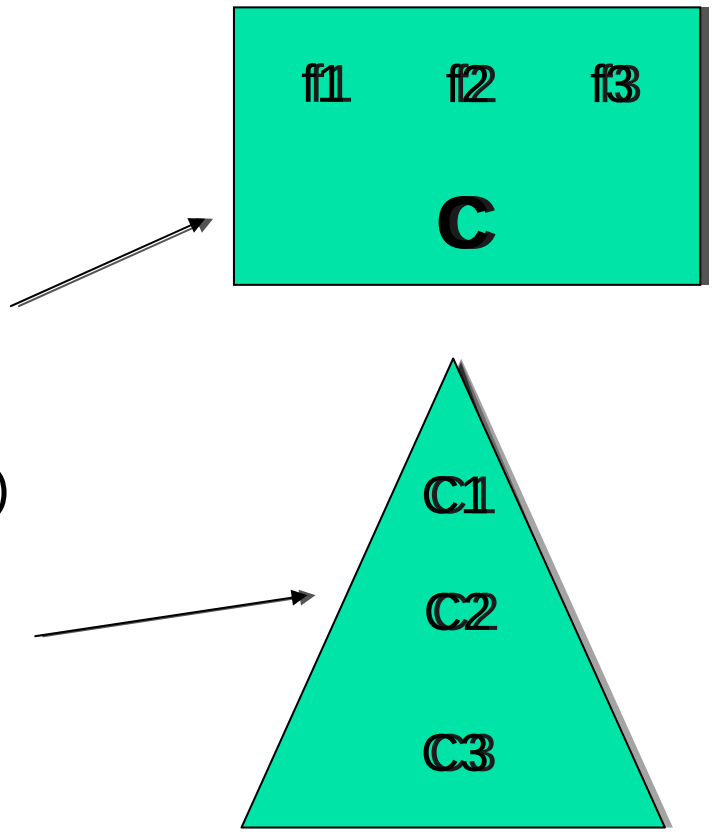
Related work: pervasive incremental SAT (or PISAT)

- Previous approaches to solving related SAT objectives were based on *incremental* search.
- In most of the previous works [Sakallah et al, Strichman 2001, the problem is as follows:
- Pervasive clauses are logical consequences of the yellow region



Related work: Fully incremental SAT (or FISAT)

- All learned clauses are pervasive and re-usable as temporary clauses are dealt with within internal SAT search
  - [Goldberg,Novikov 2001] consider the same SSAT problem (but solve tem iteratively&incrementally)
  - [Een, Sorensson 2003] consider related problems for bounded model checking domain (one safety property at a time)



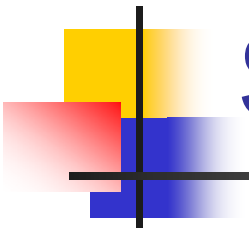
What's new?
SSAT vs PISAT and FISAT

- SSAT is oriented towards simultaneous solving while PISAT and FISAT are oriented towards *incremental* but still iterative solving
  - Therefore PISAT and FISAT do not conform to the "all watched" and "one traversal" principles of SSAT
    - The advantages of these features of SSAT will be demonstrated by experimental results (presented later on)

What's new?
SSAT vs PISAT and FISAT

- Full incrementality is a side effect of SSAT (not a goal)
- SSAT approach can be seen as orthogonal to incremental verification
  - SSAT can be used in an incremental fashion when several CNFs (with multiple objectives each) are involved
    - Will be demonstrated on simultaneous model checking of safety properties

An application of SSAT: SAT based Model Checking

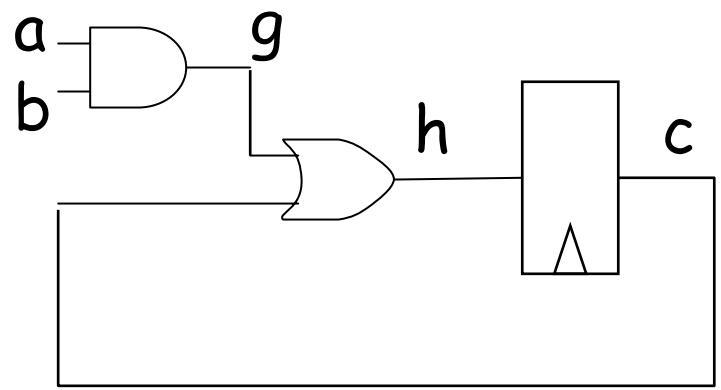


- A model is given as a set of states S, initial state relation I(S), and transition relation $TR \subseteq S \times S$
- An invariant property to check can be seen as a propositional formula P on states
- The Bounded Model Checking (BMC) problem is to check whether P is valid in the initial states and all the states reachable from them in k transition.
- If P is not violated for sufficiently large k (the diameter of the model), then it is valid (in all reachable states)

Encoding example

- A model is given as a set of propositional constraints
  - g = a&b
  - P = g+c
  - c' = h
- The property P is represented by a variable p (and the relations with the variables a,b,g,h,c)

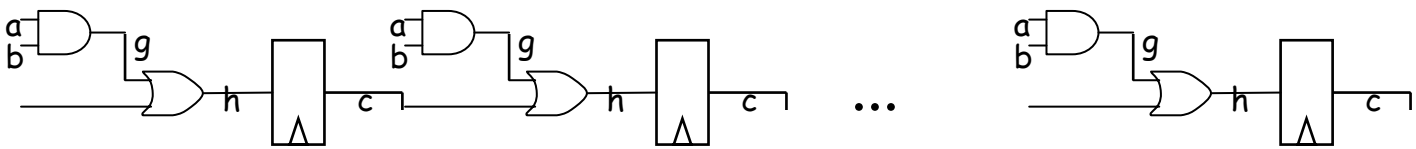
- The model



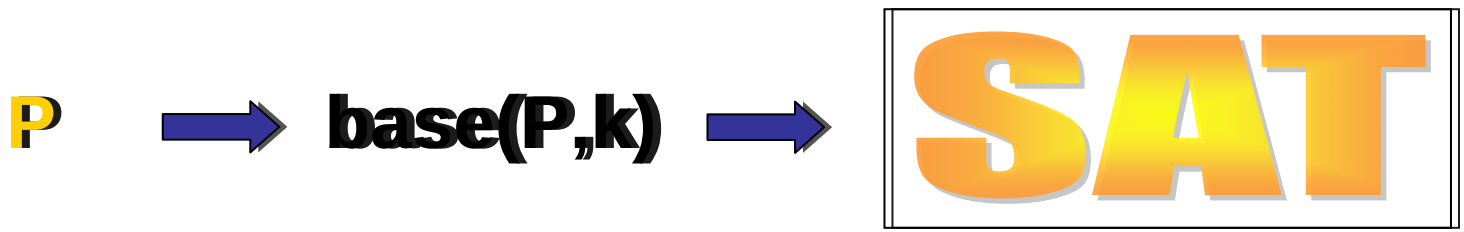
- The property: P = g & h
  - p -> g & h
  - g & h -> p

BMC for property P [Biere et al 1999]

- Unroll the model (unroll the constraints and the property)
  - Introduce variable a1, ..., ai for each time frame i; similarly for variables b, h, c, p
- Check whether the property P is falsifiable in the final state



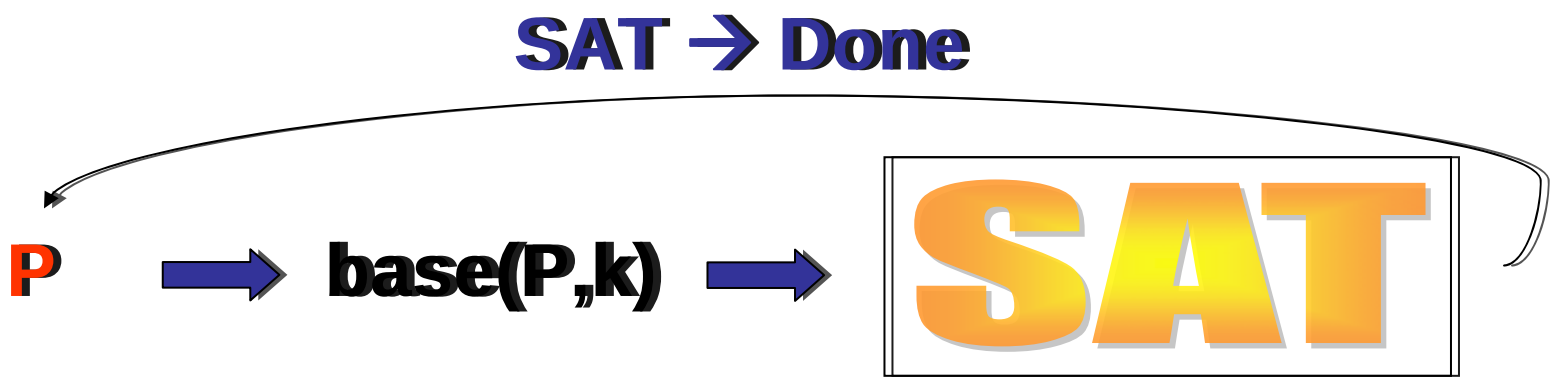
BMC for single property [Biere et al 1999]



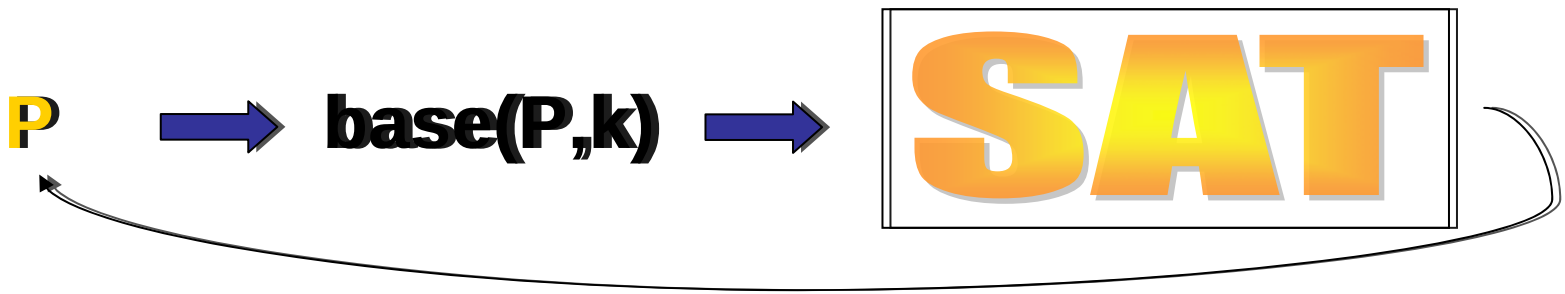
$$base(P,k) = I(s0) \wedge path(s0,\ldots,sk) \wedge P(s0) \wedge \ldots \wedge P(s[k-1]) \wedge \neg P(sk)$$

$$path(s0,\ldots,sk) = Tr(s0,s1) \wedge \ldots \wedge Tr(s[k-1],sk)$$

BMC for single property [Biere et al 1999]

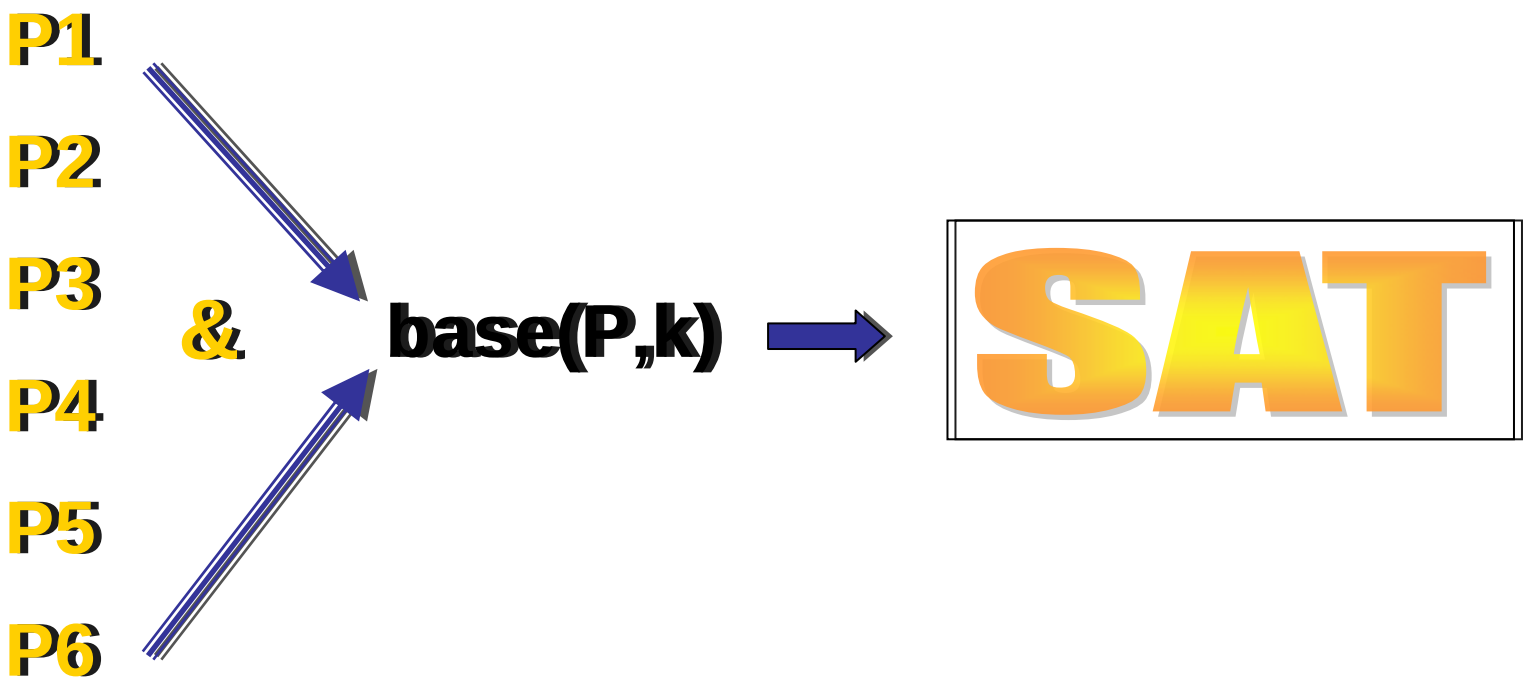


BMC for single property [Biere et al 1999]

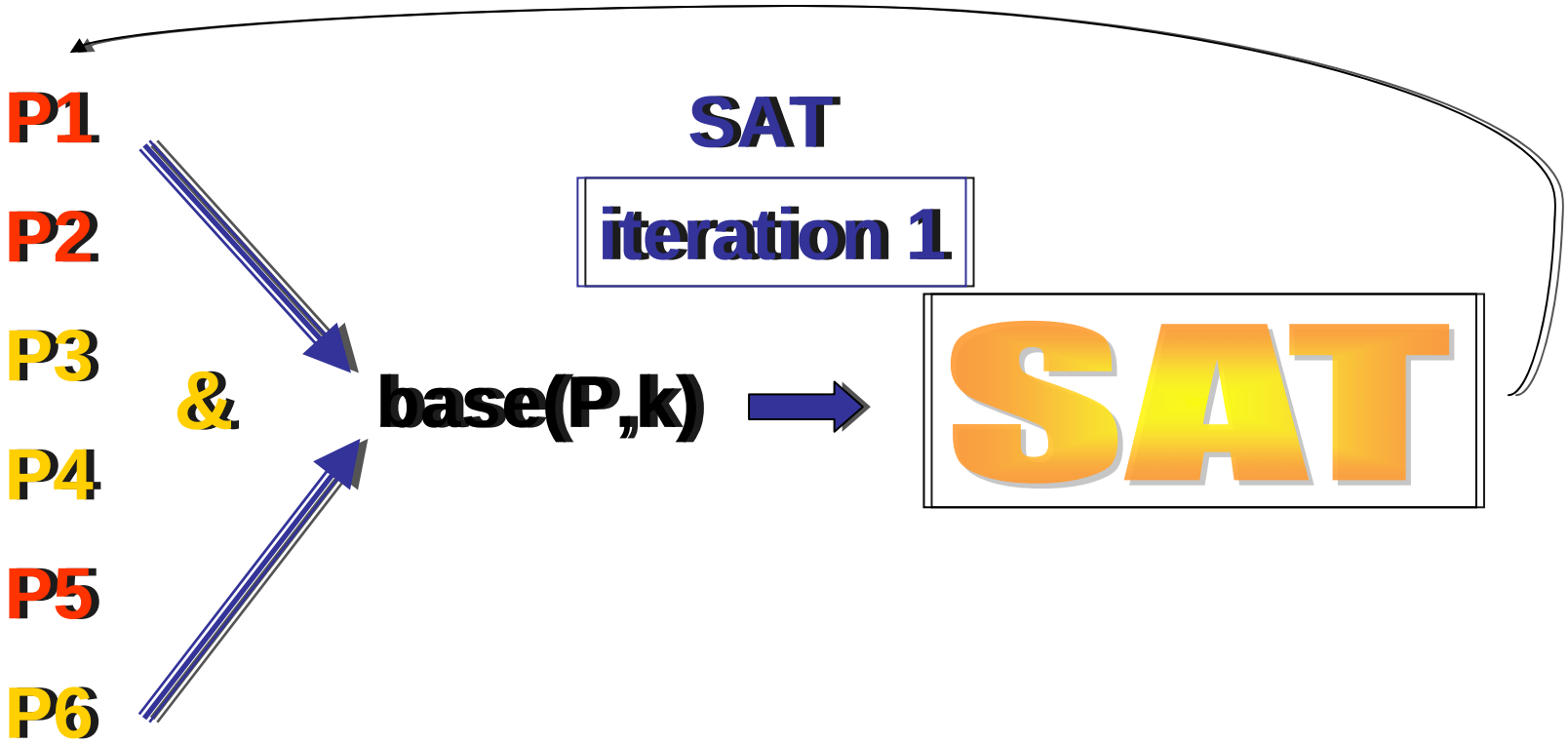


UNSAT → increase k and repeat

Simultaneous BMC – Conjunction Method [Fraer et al 2002]

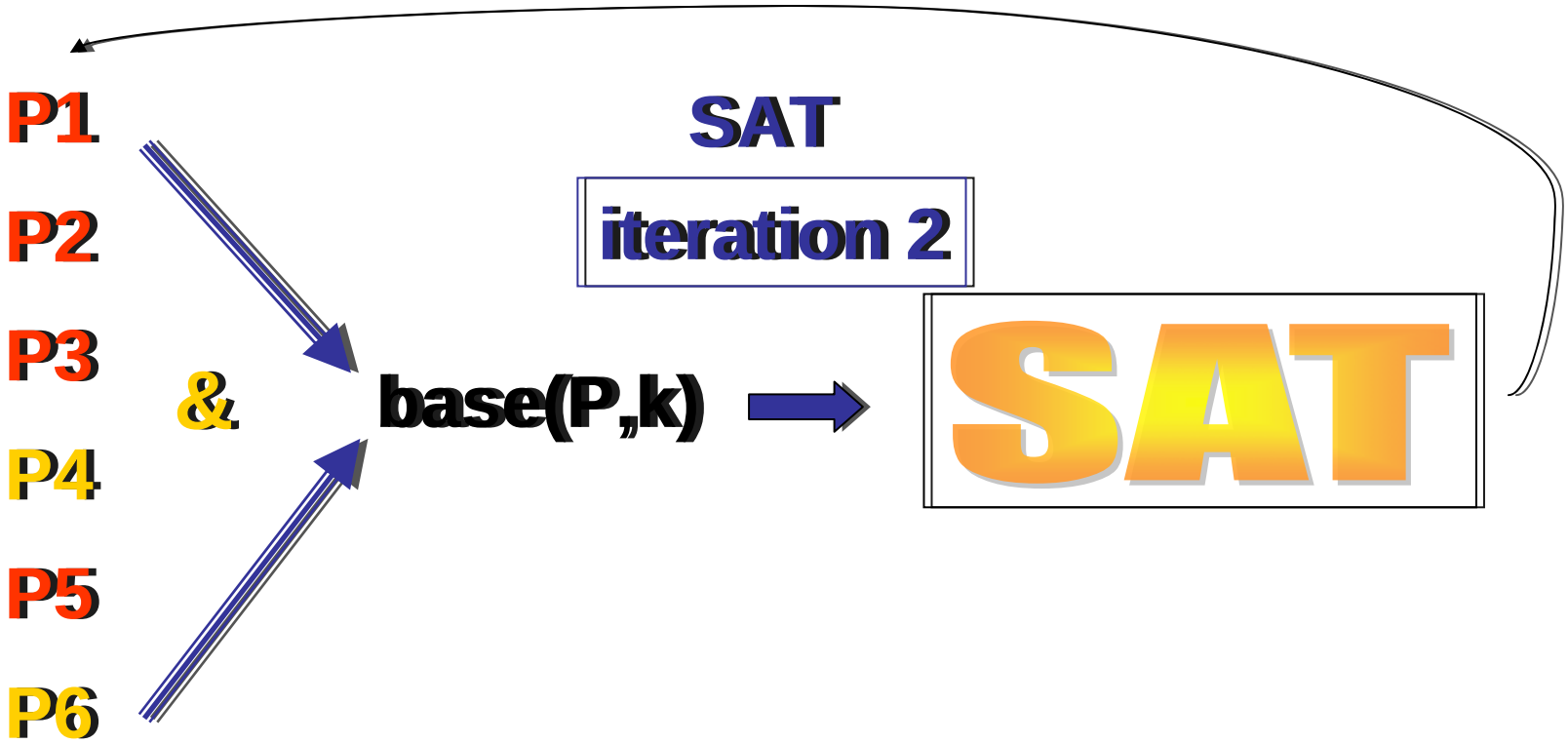


Simultaneous BMC – Conjunction Method [Fraer et al 2002]



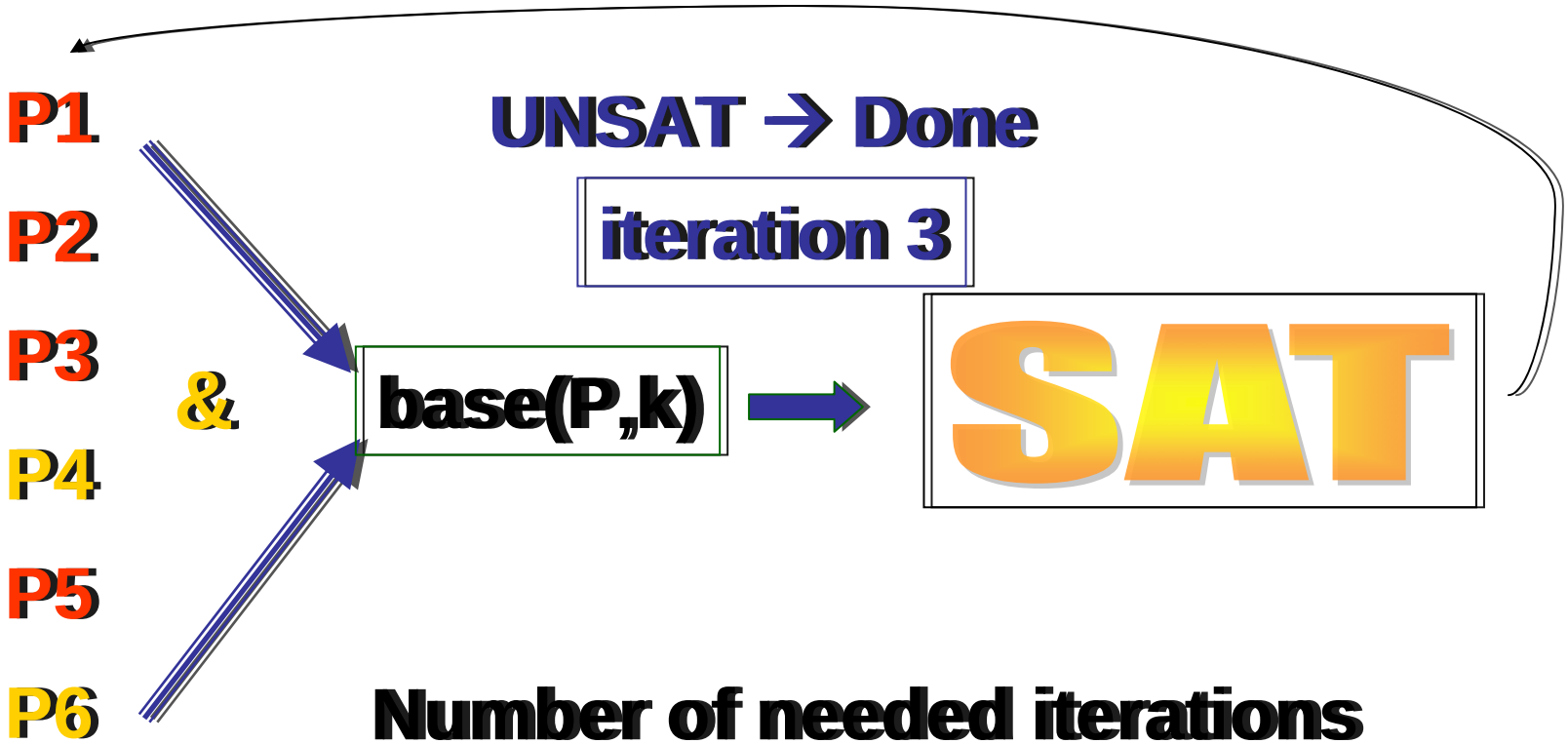
Model: {!p1, !p2, p3, p4, !p5, p6,....}

Simultaneous BMC – Conjunction Method [Fraer et al 2002]



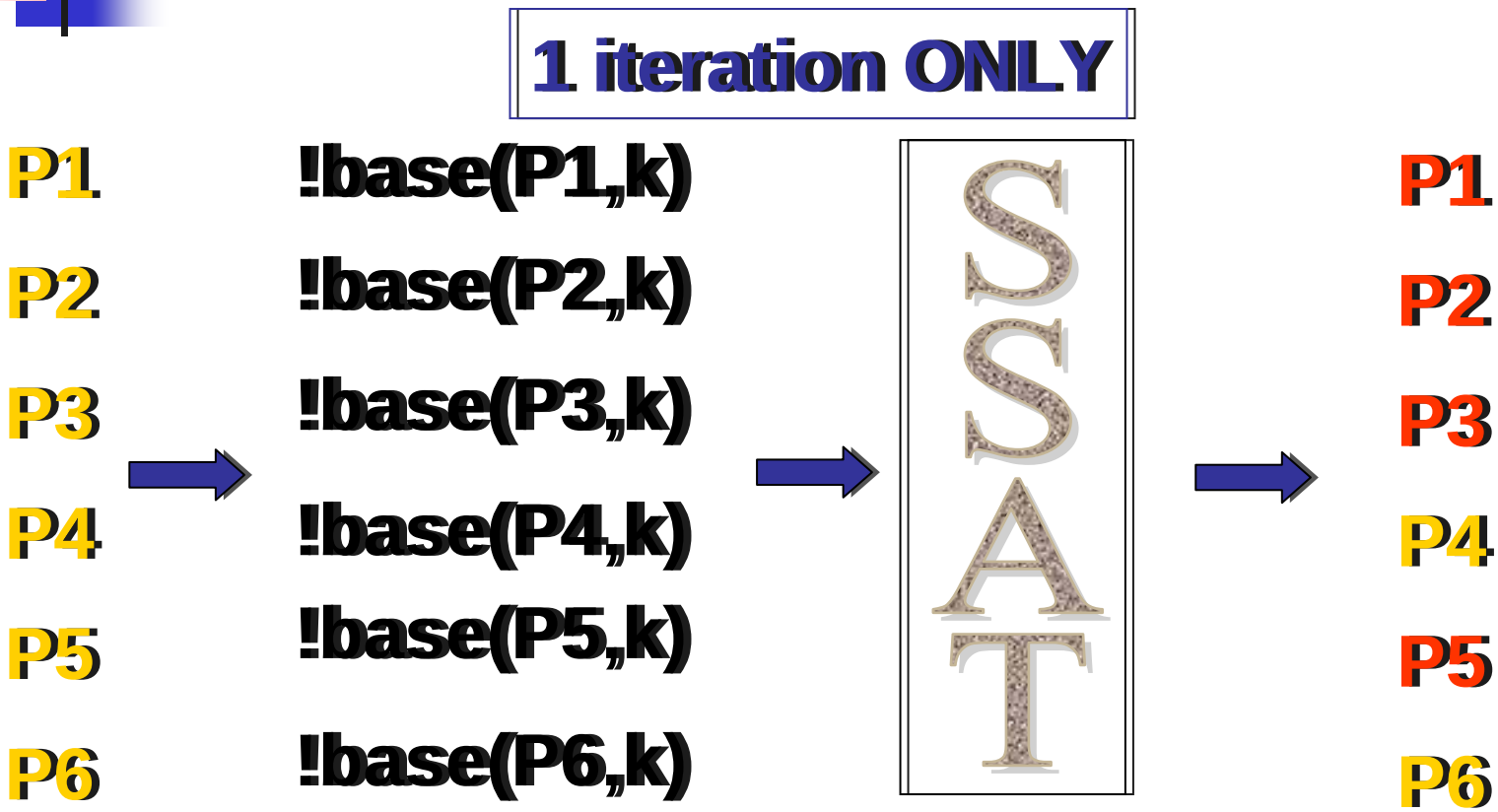
Model: {!p3, p4, p6,----}

Simultaneous BMC – Conjunction Method [Fraer et al 2002]

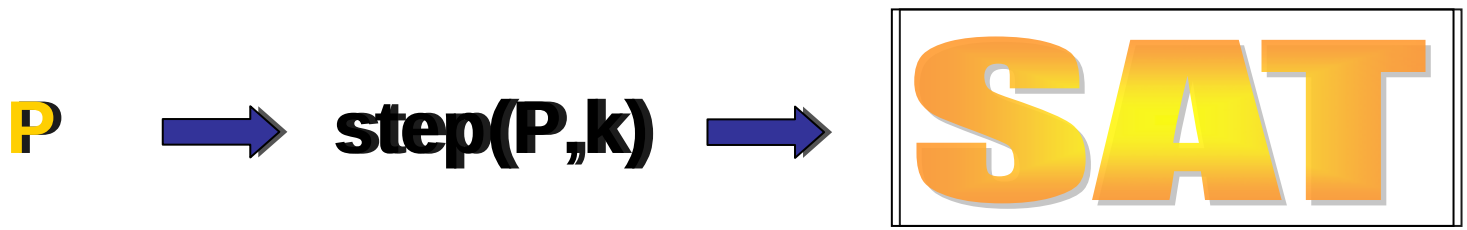


Number of needed iterations depends on the "quality" of the models

Simultaneous BMC – SSAT based



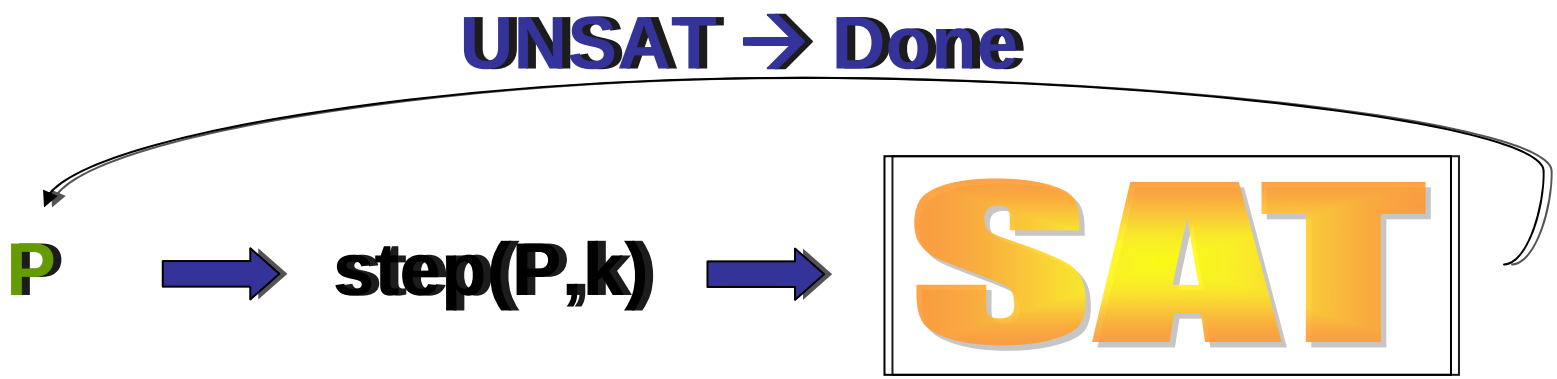
Induction step for single property [Sheeran et al 2000]



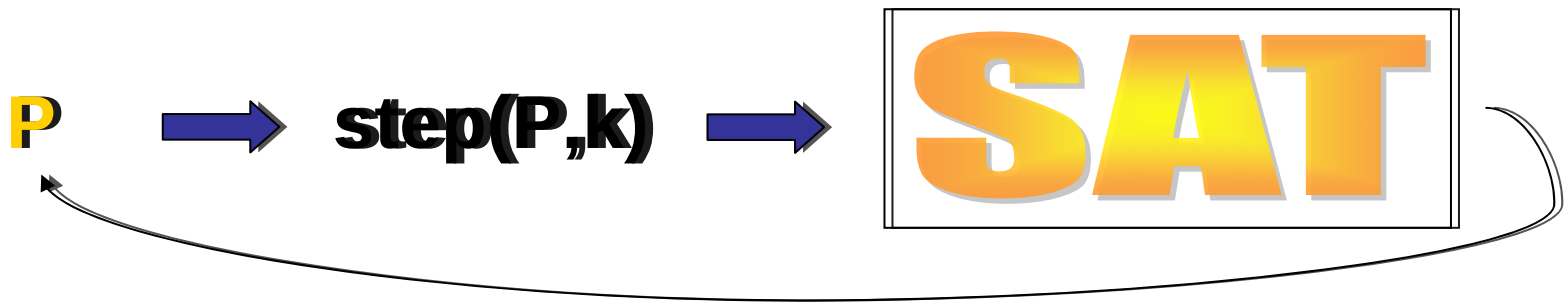
$$step(P,k) = loopFree(k+1) \wedge P(s0) \wedge \ldots \wedge P(sk) \wedge \neg P(s[k+1])$$

$$loopFree(k) = path(s0,\ldots,sk) \wedge (\bigwedge 0 \leq i < j \leq k\ (si \neq sj))$$

Induction step for single property [Sheeran et al 2000]



Induction step for single property [Sheeran et al 2000]

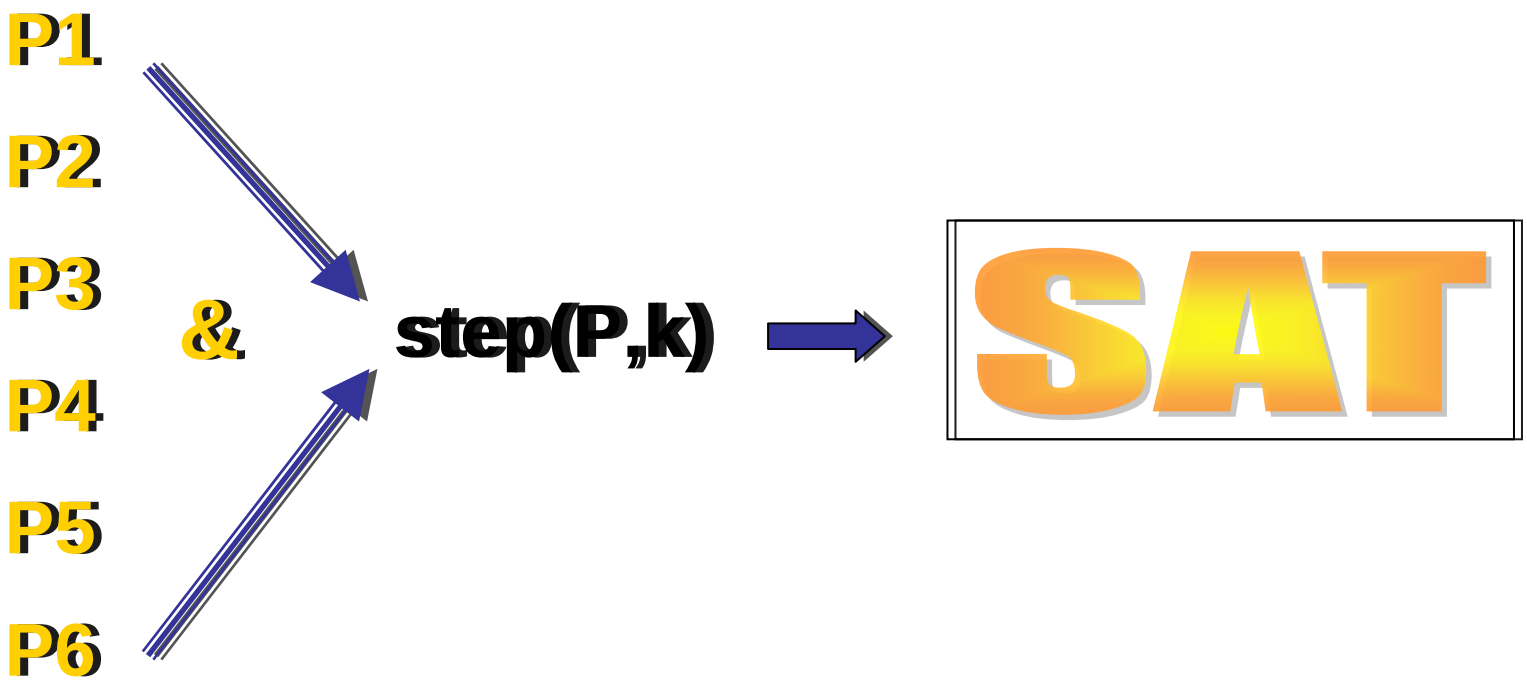


SAT → increase k and continue with base check

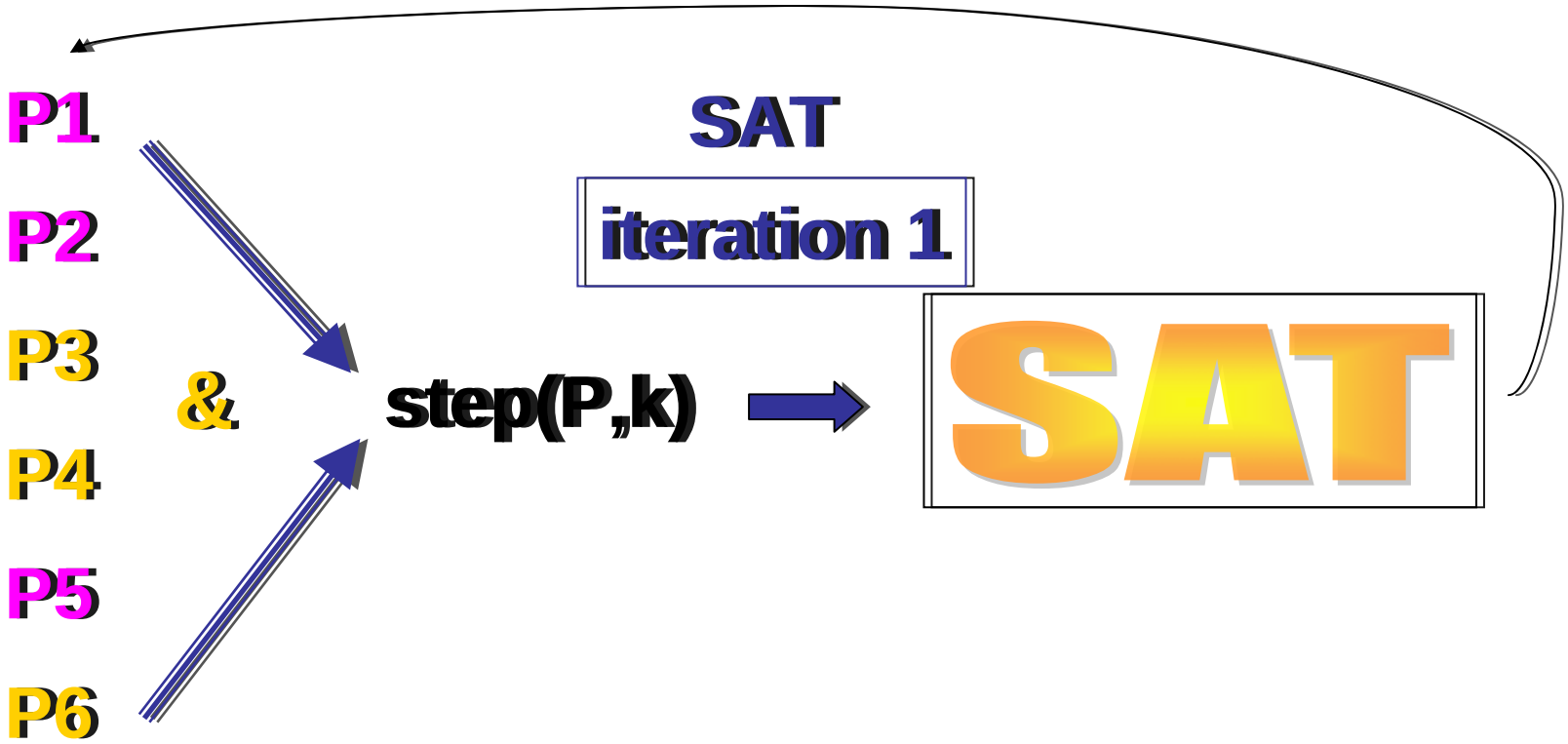
A basic version of Simultaneous Induction for U = {P1,…,Pn}

```
SIMULTANEOUS-INDUCTION(U , max_depth) {
k = 0;
    while ( k ≤ max_depth && U != ∅) {
       U = simultaneous_base(U,k); // report falsified
       properties
       If ( U != ∅)
              U = simultaneous_step(U,k); // report proved
       properties
       k++;
    }
Return U; // return unresolved properties
}
```

Simultaneous step – Conjunction Method [Fraer et al 2002]

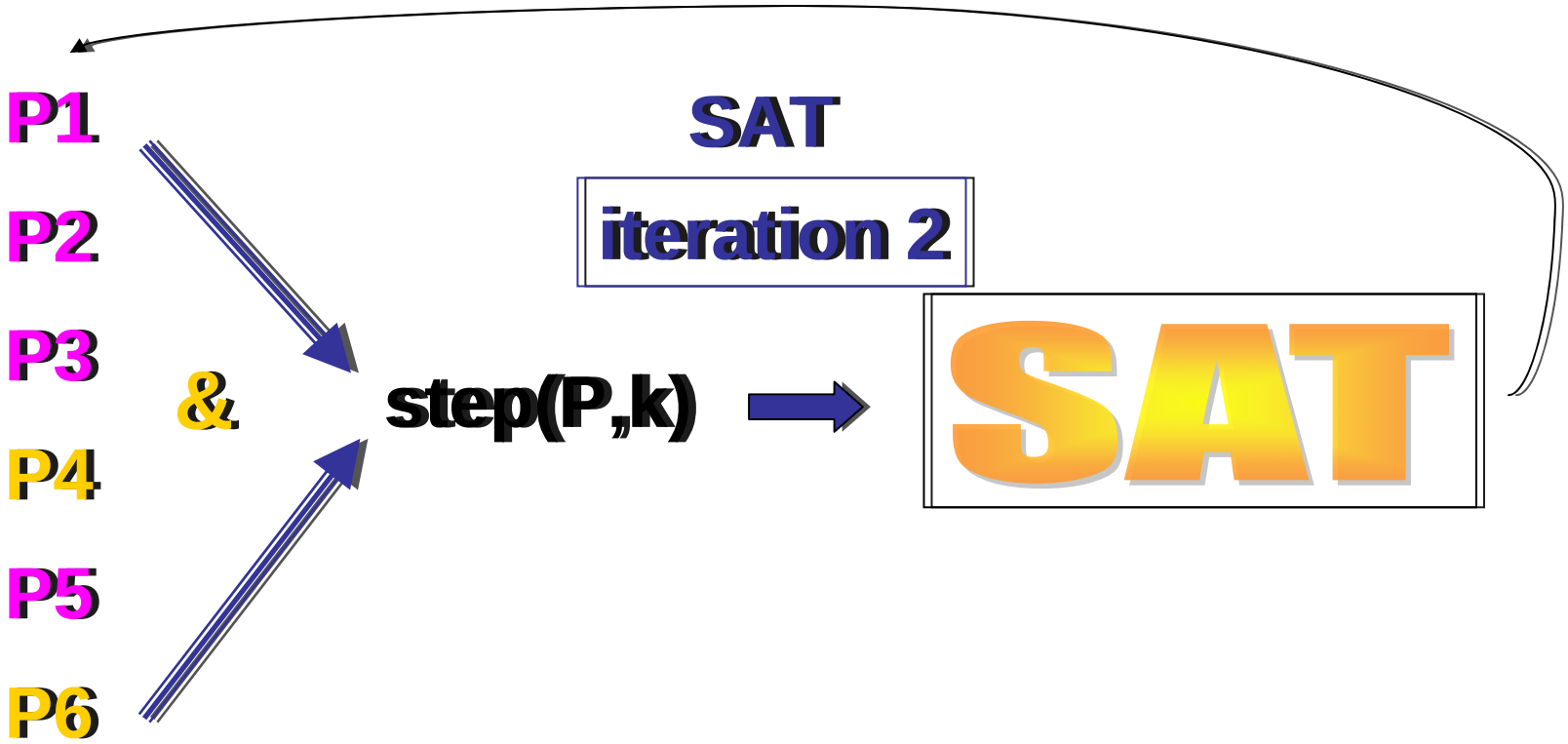


Simultaneous step – Conjunction Method [Fraer et al 2002]



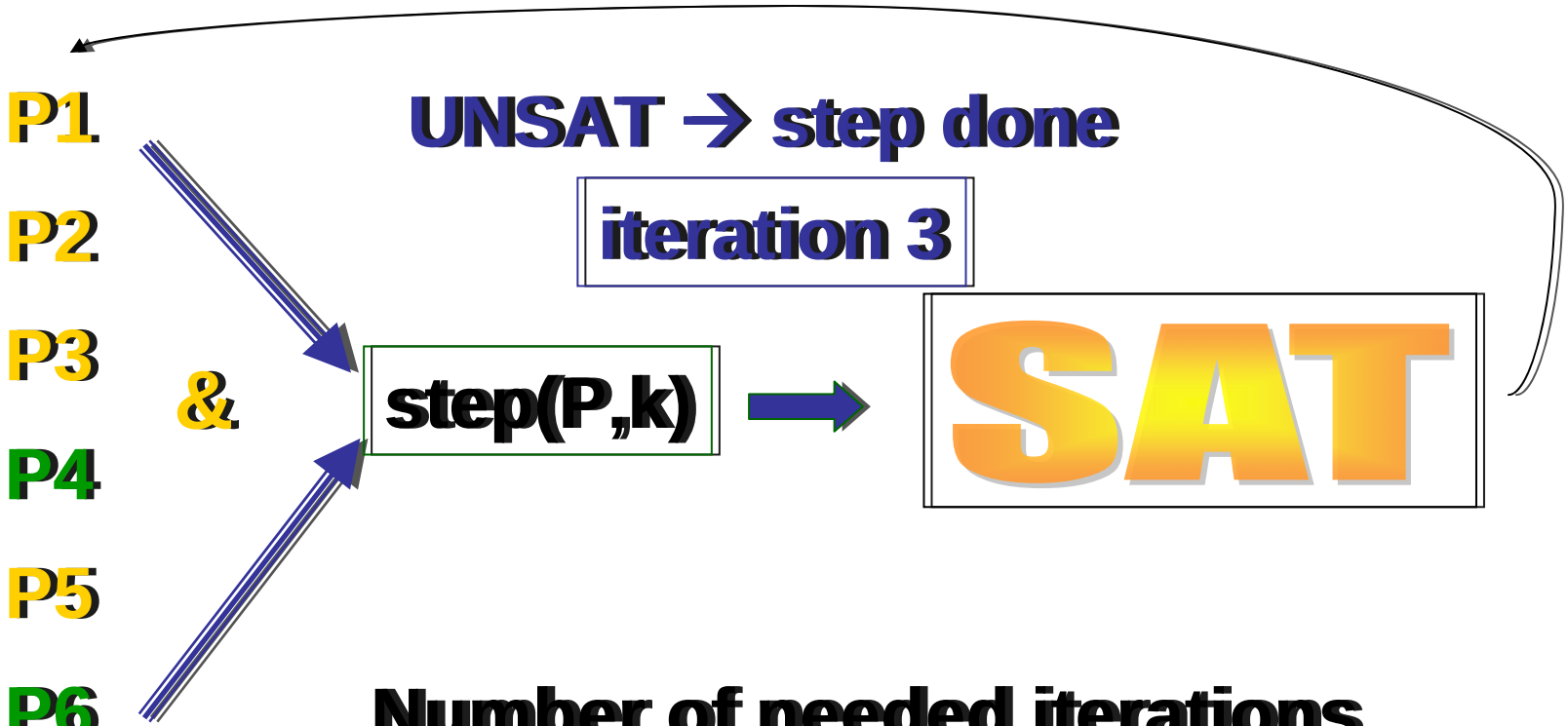
Model: {!p1, !p2, p3, p4, !p5, p6,----}

Simultaneous step – Conjunction Method [Fraer et al 2002]



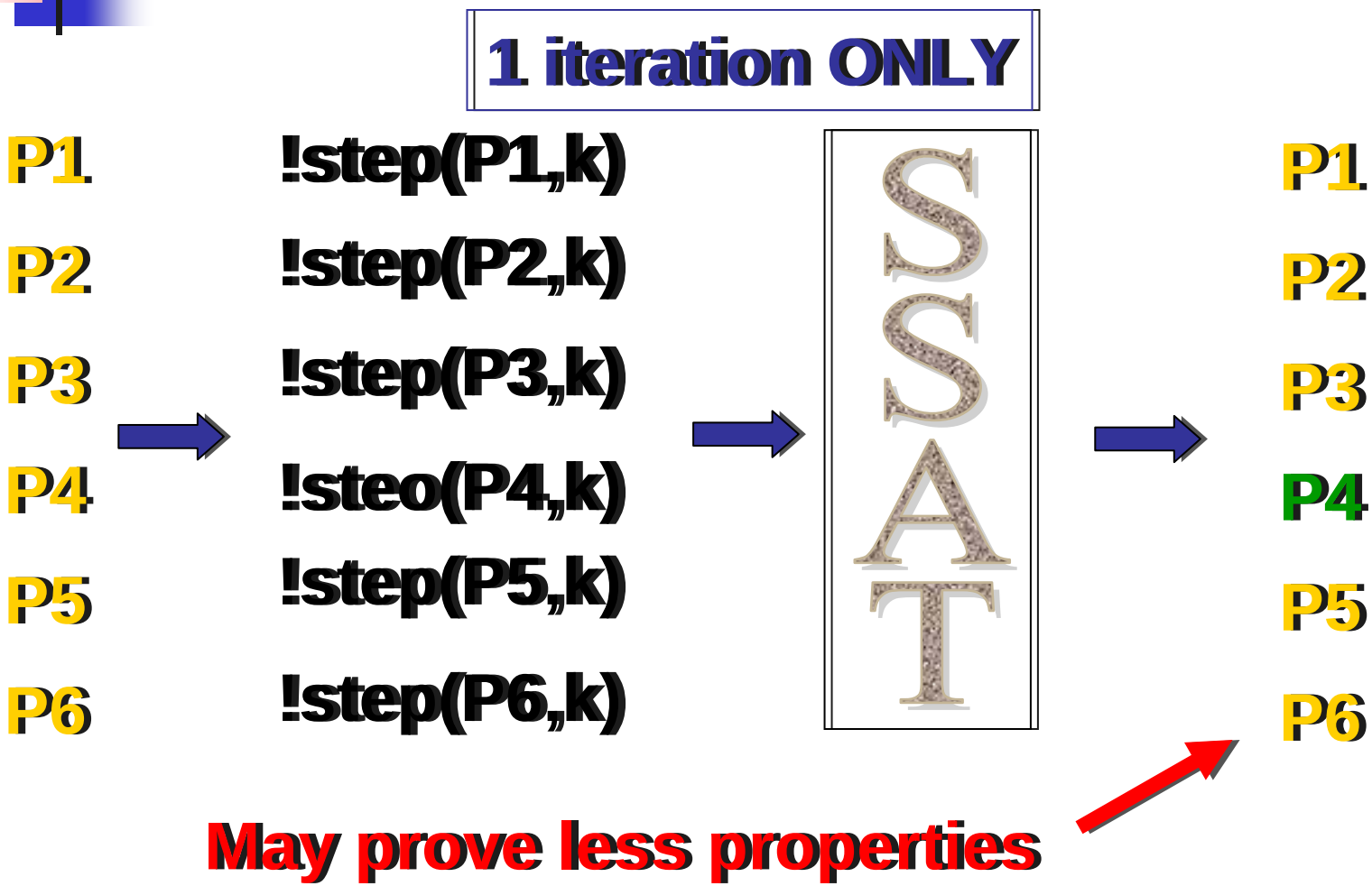
Model: {!p3, p4, p6,----}

Simultaneous step – Conjunction Method [Fraer et al 2002]

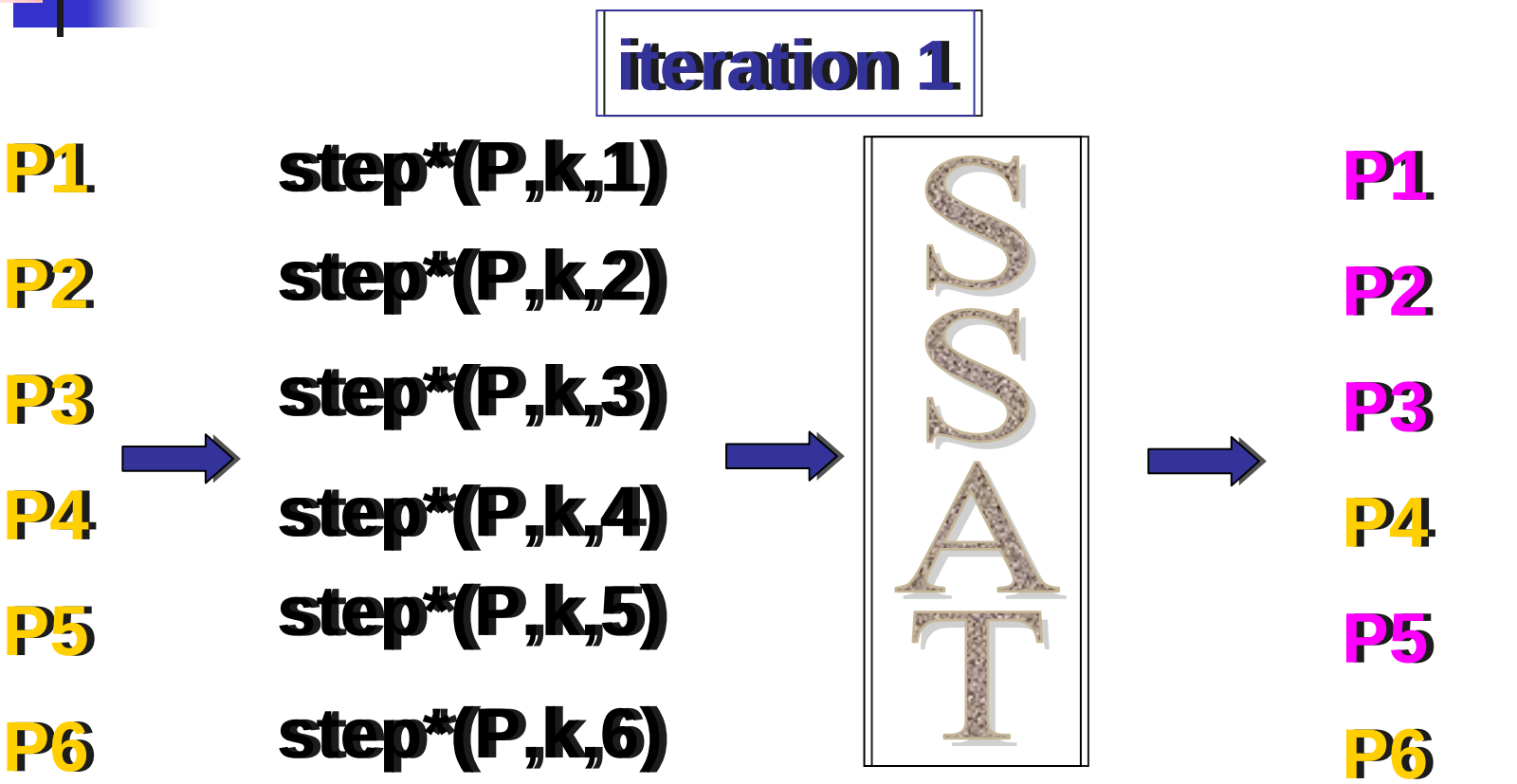


Number of needed iterations depends on the "quality" of the models

Simultaneous step – SSAT based: Version 1

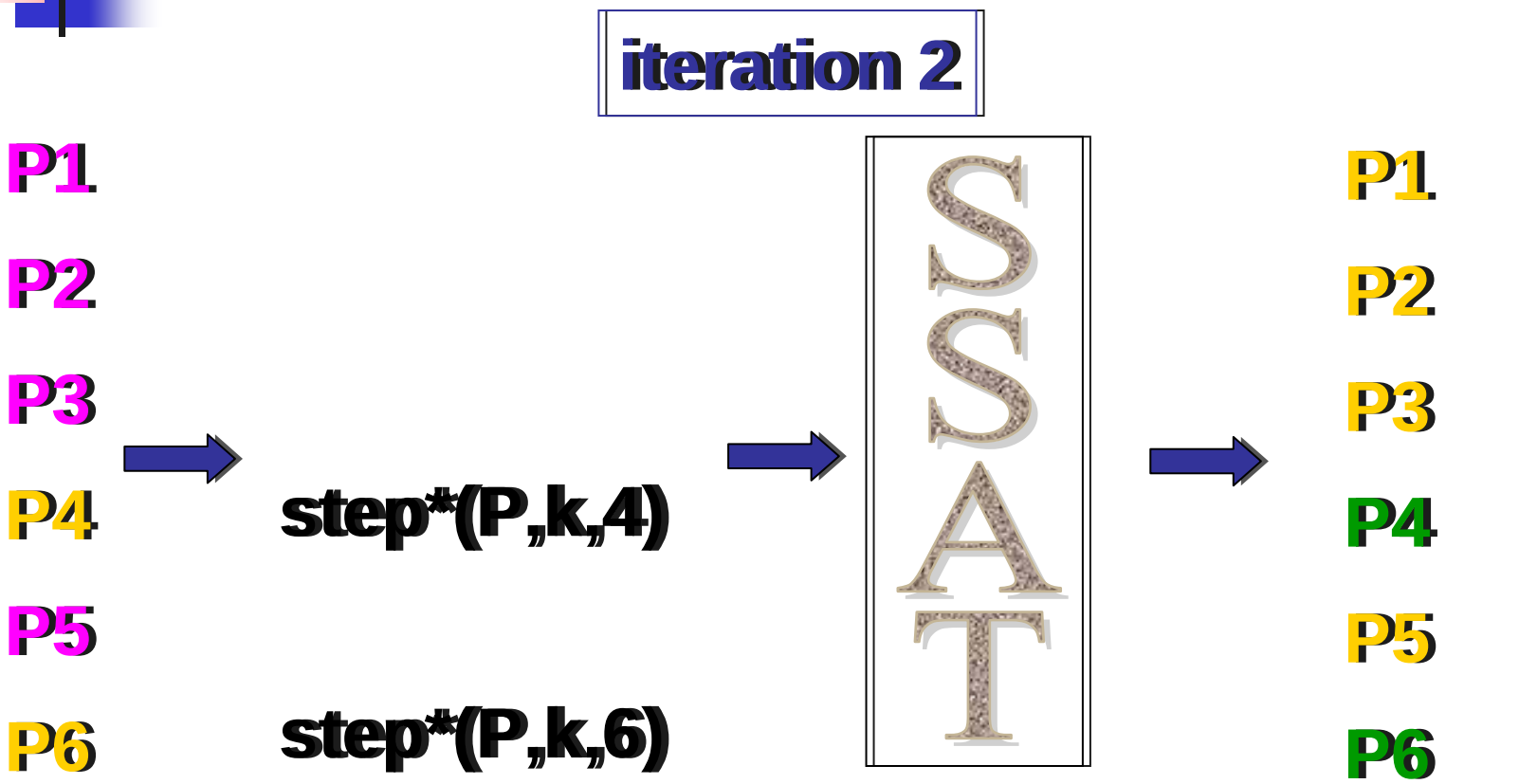


Simultaneous step – SSAT based: Version 2 -- Hybrid



$$step^*(P,k,l) = \ldots \wedge P(s0) \wedge \ldots \wedge P(sk) \rightarrow P\,i\,(s[k+1])$$

Simultaneous step – SSAT based: Version 1



$$step^*(P,k,l) = \ldots \wedge P(s0) \wedge \ldots \wedge P(sk) \rightarrow P\,i\,(s[k+1])$$

Incremental simultaneous BMC and Induction

- All schemes can be made “double incremental” :
  - pervasive (or all) conflict clauses can be reused in every iteration at each depth
  - Pervasive (or all) conflict clauses can be transferred and re-used at higher depths
- In PISAT approach, “temporal” variables involved in the base and step formulas can be removed
- In FISAT, they must be kept
  - Not a significant overhead in general, but may become a significant performance issue in some cases/algorithms

Propositional benchmark (base): PISAT vs FISAT vs SSAT

BMC depth	POs	gates	inputs	literals	clauses	PISAT	GN+	SSAT
6	543	98288	25383	93784	254145	174.36	9.28	2.42
7	494	113938	28885	108488	295157	245.51	5.41	4.68
8	473	132372	33352	125745	342993	210.47	8.14	3.96
9	450	150565	37454	142720	390432	316.93	2.61	2.61
10	450	170016	42072	160938	440968	305.79	11.79	6
11	435	189670	46529	179233	492157	508.97	14.61	11.6
12	418	209885	51380	198212	544750	364.76	7.68	6.69
13	417	229883	55880	216769	596763	576.3	5.72	5.71
14	417	250285	60745	235896	649809	424.04	11.14	11.38
15	415	270393	65243	254571	702148	686.75	7.96	8.05
Total						3813.88	84.34	63.1

Propositional benchmark (step): PISAT vs FISAT vs SSAT

Step depth	POs	Gates	Inputs	Literals	Clauses	PISAT	GN+	SSAT
1	543	52786	8026	45811	132490	87.42	30.3	1.99
2	433	73274	9024	62397	184397	117.71	44.6	2.2
3	433	94927	10065	79932	239277	170.86	62.13	3.26
4	433	117160	11128	97964	295687	230.31	78.97	3.92
5	433	140251	12196	117095	355143	291.87	97.73	5.03
6	384	161723	13135	134894	410561	323.46	109.5	15.93
7	260	182905	14182	152888	465867	273.91	117.62	10.36
8	236	204098	15210	170834	521100	292.09	117.87	8.56
9	236	225355	16241	188819	576470	321.43	131.28	11.03
10	221	246438	17261	206653	631386	340.79	139.34	9.24
Total						2449.85	929.34	71.52

Simultaneous induction benchmark (various methods)

# properties	non-incr conj	double incr conj	double incr GN+	incr SSAT	double incr SSAT	double incr Hybrid Ind
9	189.57	72.67	66.65	35.05	29.84	29.32
9	200.13	73.23	61.72	40.46	27.76	29.09
9	222.29	66.11	67.17	35.84	26.06	27.83
9	246.51	67.85	62.33	37.22	28.24	29.5
9	253	68.14	59.55	39.04	28.66	30.13
9	215.09	70.25	60.5	35.52	26.7	27.86
Total (sec)	**1326.59**	**418.25**	**377.92**	**223.13**	**167.26**	**173.73**

Conflicts, globally true, simultaneous falsification data

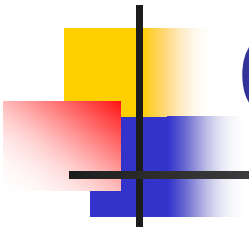
Number of POs	PISAT pervasive conflicts	PISAT all conflicts	SSAT all conflicts	SSAT globally true	SSAT models	SSAT falsifiable POs
9×45	6170	35225	13615	126	21	189
9×45	8549	40680	15709	135	21	189
9×45	8258	37814	14206	135	21	189
9×45	8488	39945	14276	135	21	189
9×45	7056	35370	14251	135	21	189
9×45	6968	13257	13257	135	21	189

Simultaneous BMC benchmark (various methods)

property count	BMC depth	non-incr conj	double incr conj	incr-GN+	double incr GN+	incr SSAT	double incr SSAT
32	50	32.95	5.4	536.3	8.31	533.87	8.14
32	50	32.85	5.46	543.15	8.42	534.85	8.15
3	50	318.87	108.54	3041.72	46.17	3064.57	46.05
3	50	360.67	464.32	3760.78	210.15	3747.52	210.45
3	50	310.64	367.93	3653.52	50.23	3612.3	50.06
3	50	242.4	231.59	3337.25	199.96	3330.65	199.46
8	50	78.8	30.69	681.68	27.46	685.82	27.18
8	50	78.14	30.77	680.2	26.97	682.88	26.91
8	50	18.53	3.28	162.84	9.98	157.66	10.03
8	50	18.46	3.35	157.25	10.23	157.69	10.18
543	15	139.82	41.18	51.26	32.78	36.08	18.91
172	30	145.75	52.13	76.83	40.5	63.59	28.06
1035	3	2478.61	406.2	1743.56	1618.58	386.93	229.28
Total run times:		4256.49	1750.84	18426.34	2289.74	16994.41	872.86

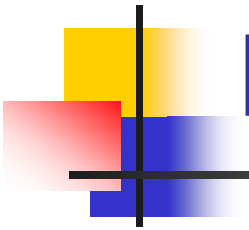
Simultaneous induction benchmark: basic schemes

PO count/ false/ Valid	depth	(1) non-incr conj	(2) double incr conj	(3) double incr SSAT	(4) double incr Hybrid	Speedup (3) vs (2)	Speedup (4) vs (2)	speedup (4) vs (1)
172/106/65	30	2777.54	1718.51	200.69	237.66	8.56	7.23	11.69
543/128/(275:214)	30	3978.52	3287.89	312.08	233.97	10.54	14.05	17.00
100/0/(73:41)	10	18040.21	6390.95	830.61	1384.08	7.69	4.62	13.03
249/32/(70:64)	14	31338.58	7730.62	3710.35	2148.58	2.08	3.60	14.59



Conclusions

- We presented a simultaneous propositional satisfiability algorithm which outperforms current BKM of solving related SAT tasks incrementally
- We proposed SSAT based BMC and induction schemes for simultaneous model-checking of invariant properties which is an order of magnitude faster than current state of the art



Future work

- It should be possible to explore and exploit the “all watched” and “one traversal” advantages of SSAT even further
- We believe SSAT can be useful in other applications as well where incremental SAT is successful