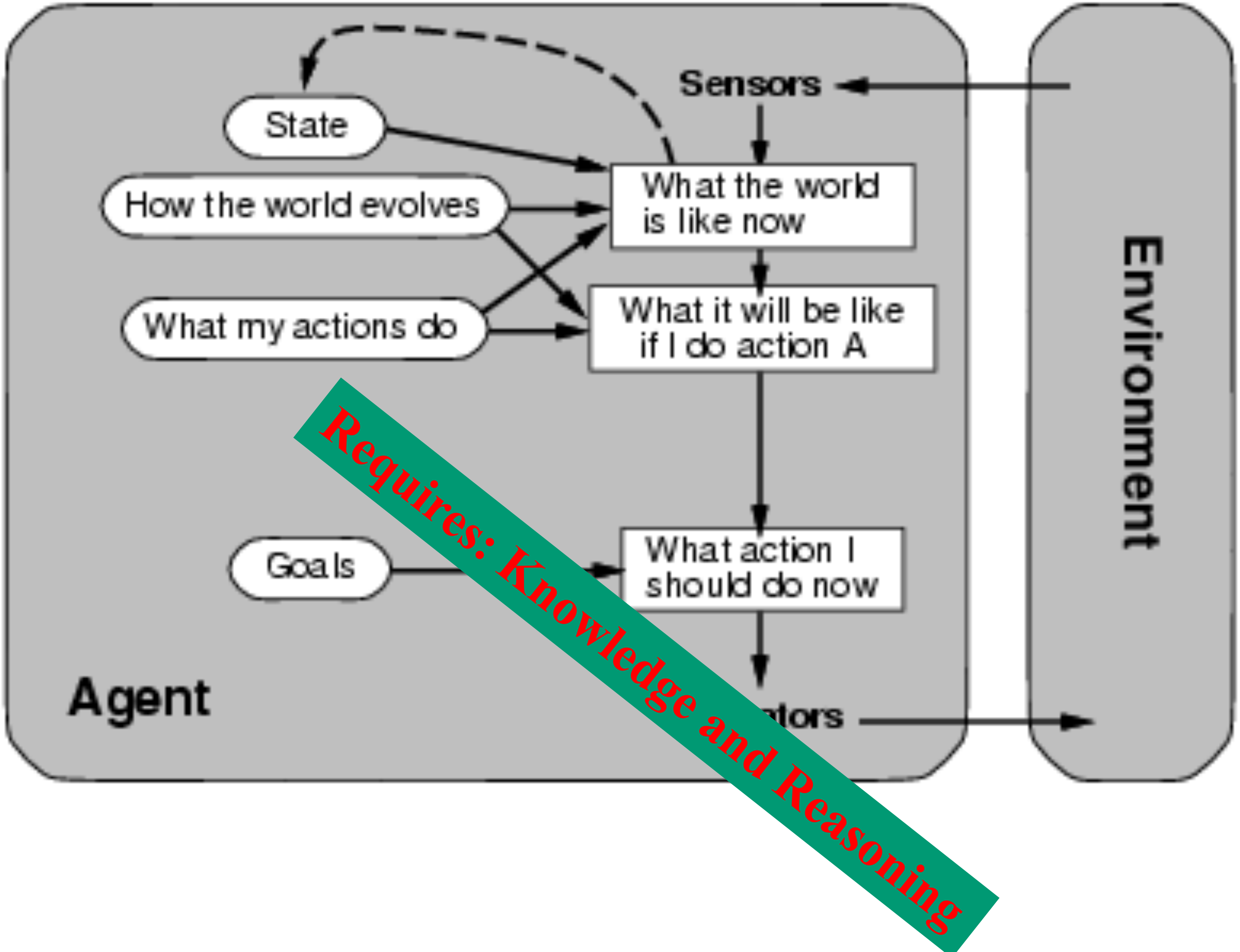
CS 4700: Foundations of Artificial Intelligence

Bart Selman
selman@cs.cornell.edu

Logical Agents --- Intro Knowledge Representation & SAT encodings

R&N: Chapter 7

A Model-Based Agent



Knowledge and Reasoning

Knowledge and Reasoning:

humans are very good at acquiring new information by combining raw knowledge, experience with reasoning.

AI-slogan: "Knowledge is power" (or "Data is power"?)

Examples:

Medical diagnosis --- physician diagnosing a patient infers what disease, based on the knowledge he/she acquired as a student, textbooks, prior cases

Common sense knowledge / reasoning --- common everyday assumptions / inferences. e.g., (1) "lecture starts at four" infer pm not am; (2) when traveling, I assume there is some way to get from the airport to the hotel.

Logical agents:

Agents with some representation of the complex knowledge about the world / its environment, and uses inference to derive new information from that knowledge combined with new inputs (e.g. via perception).

Key issues:

1- Representation of knowledge

What form? Meaning / semantics?

2- Reasoning and inference processes

Efficiency.

Knowledge-base Agents

Key issues:

- Representation of knowledge → knowledge base
- Reasoning processes → inference/reasoning

Knowledge base = set of sentences in a formal language representing facts about the world (*)

(*) called Knowledge Representation (KR) language

Knowledge Representation language candidate:
logical language (propositional / first-order)
combined with a logical inference mechanism

Why not use natural language (e.g. English)?

We want clear syntax & semantics (well-defined meaning), and, mechanism to infer new information.
Soln.: Use a formal language.

More Concrete: Propositional Logic

Syntax: build sentences from atomic propositions, using connectives $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$.
(and / or / not / implies / equivalence (biconditional))

E.g.: $((\neg P) \vee (Q \wedge R)) \Rightarrow S$

Semantics

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$	$\neg P \vee Q$
False	False	True	False	False	True	True	True
False	True	True	False	True	True	False	True
True	False	False	False	True	False	False	False
True	True	False	True	True	True	True	True

Note: $\Rightarrow$ somewhat counterintuitive.

What's the truth value of "5 is even implies Sam is smart"?

True!

Satisfiability

9

Propositional Satisfiability problem

Satifiability (SAT): Given a formula in propositional calculus, is there a model (i.e., a satisfying interpretation, an assignment to its variables) making it true?

We consider clausal form, e.g.:

$$(a \vee \neg b \vee \neg c) \text{ AND } (b \vee \neg c) \text{ AND } (a \vee c)$$

2^n possible assignments

SAT: prototypical hard combinatorial search and reasoning problem. Problem is NP-Complete. (Cook 1971)

Surprising "power" of SAT for encoding computational problems

Modern SAT Solvers use this langauge.

SAT Solvers

Modern Satisfiability (SAT) solvers operating on the clausal form are surprisingly *much* more efficient than other logic-based approaches (e.g., pure resolution that will be discussed later)

The SAT solvers treat the set of clauses as a set of constraints (disjunctions) on Boolean variables, i.e., a CSP problem! Current solvers are very powerful. Can handle 1 Million+ variables and several millions of clauses.

Systematic: Davis Putnam (DPLL) + *series of improvements*
Stochastic local search: WalkSAT (issue?)

See R&N 7.6. "Ironically," we are back to semantic model checking, but way more clever than basic truth assignment enumeration (exponentially faster)!

Satisfiability
as an Encoding Language

SAT Translation of Graph Coloring

Variables

NxK vars

$C_{i,k}$ node i has color k

Total # colors: K. Total # nodes: N.

At least one of K colors per node i :

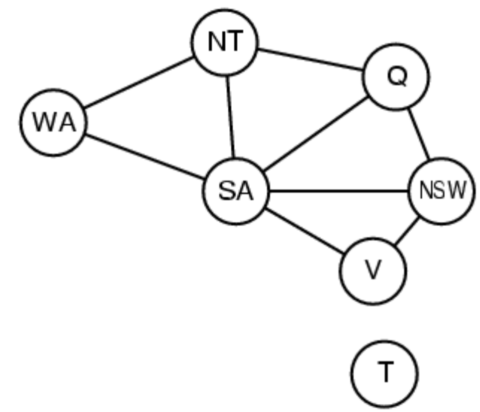
$$(C_{i,1} \vee C_{i,2} \vee C_{i,3} \vee \ldots \vee C_{i,K})$$

At most one color per node i :

$$(\sim C_{i,k} \vee \sim C_{i,k'}) \quad \text{for all } k =/= k'$$

If node i and node j (=/= i) share an edge, need to have different colors:

$$(\sim C_{i,l} \vee \sim C_{j,l}) \quad \text{for all } 1 <= l <= K$$



Note: Translation from "problem" into SAT. Reverse of usual translation to show NP-completeness.

Works also for (easy) polytime problems!

n x n square

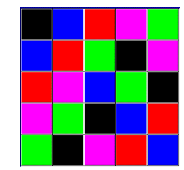
Encoding Latin Square Problems in Propositional Logic

Variables: n^3

x_{ijk} $cell\ i,j\ has\ color\ k;\ \ i,j,k = 1,2,\ldots,n.$

$x_{ijk} \in \{0,1\}$

Each variables represents a color assigned to a cell.



Clauses: $O(n^4)$

- Some color must be assigned to each cell (clause of length n); n^2

$$\forall_{ij}(x_{ij1} \vee x_{ij2} \ldots x_{ijn})$$

- No color is repeated in the same row (sets of negative binary clauses); n(n-1)/2 n n

(example Row i Color k)

$$\forall_{ik}(\neg x_{i1k} \vee \neg x_{i2k}) \wedge (\neg x_{i1k} \vee \neg x_{i3k}) \ldots (\neg x_{i1k} \vee \neg x_{ink}) \cdots (\neg x_{i(n-1)k} \vee \neg x_{ink})$$

- No color is repeated in the same column (sets of negative binary clauses);

n(n-1)/2 n n

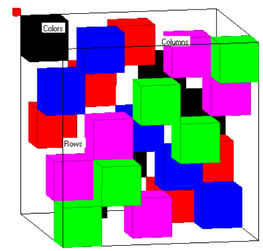
(example Colum j Color k)

$$\forall_{jk}(\neg x_{1jk} \vee \neg x_{2jk}) \wedge (\neg x_{1jk} \vee \neg x_{3jk}) \ldots (\neg x_{1jk} \vee \neg x_{njk}) \cdots (\neg x_{(n-1)jk} \vee \neg x_{njk})$$

3D Encoding or Full Encoding

This encoding is based on the cubic representation of the quasigroup: each line of the cube contains exactly one true variable;

Variables: $O(n^4)$ Same as 2D encoding.



Clauses:

•Same as the 2 D encoding plus:

- Each color must appear at least once in each row;

$$\forall_{ik}(x_{i1k} \vee x_{i2k} \ldots x_{ink})$$

- Each color must appear at least once in each column;

$$\forall_{jk}(x_{1jk} \vee x_{2jk} \ldots x_{njk})$$

- No two colors are assigned to the same cell;

$$\forall_{ij}(\neg x_{ij1} \vee \neg x_{ij2}) \wedge (\neg x_{ij1} \vee \neg x_{ij3}) \ldots (\neg x_{ij1} \vee \neg x_{ijn})$$
$$\ldots(\neg x_{ijn} \vee \neg x_{ij(n-1)})$$

Dimacs format

At the top of the file is a simple header.

p cnf <variables> <clauses>

Each variable should be assigned an integer index. Start at 1, as 0 is used to indicate the end of a clause. The positive integer a positive literal, whereas a negative interger represents a negative literal.

Example

-1 7 0 → $(\neg x1 \vee x7)$

Extended Latin Square 2x2

```
p cnf 8 24
-1 -2 0
-3 -4 0
-5 -6 0
-7 -8 0
```

A cell gets at most a color

```
-1 -5 0
-2 -6 0
-3 -7 0
-4 -8 0
```

No repetition of color in a column

```
-1 -3 0
-2 -4 0
-5 -7 0
-6 -8 0
```

No repetition of color in a row

```
1 2 0
3 4 0
5 6 0
7 8 0
```

A cell gets a color

```
1 5 0
2 6 0
3 7 0
4 8 0
```

A given color goes in each column

```
1 3 0
2 4 0
5 7 0
6 8 0
```

A given color goes in each row

order 2

-1	-1
-1	-1

1/2	3/4
5/6	7/8

1 – cell 11 is red
2 – cell 11 is green
3 – cell 12 is red
4 – cell 12 is green
5 – cell 21 is red
6 – cell 21 is green
7 – cell 22 is red
8 – cell 22 is green

Underlying Latin Square structure characterizes many real world applications

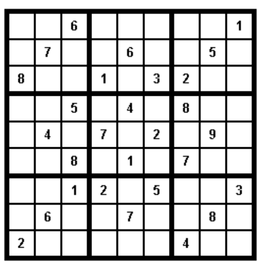
teams

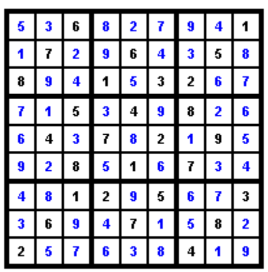
teams	.1	2	3	4	5	
•1	X	•R1	•R4	•R2	•R5	•R3
•2	•R1	X	•R2	•R5	•R3	•R4
•3	•R4	•R2	X	•R3	•R1	•R5
•4	•R2	•R5	•R3	X	•R4	•R1
•5	•R5	•R3	•R1	•R4	X	•R2
•6	•R3	•R4	•R5	•R1	•R2	X

Scheduling and timetabling

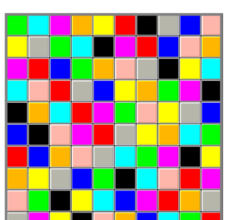
A	D	E	B	C
C	B	A	E	D
D	C	B	A	E
E	A	C	D	B
B	E	D	C	A

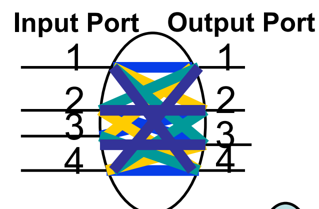
Design of Scientific Experiments

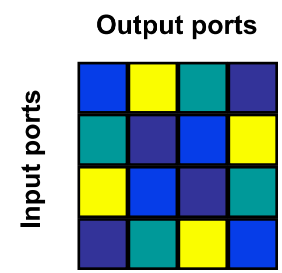




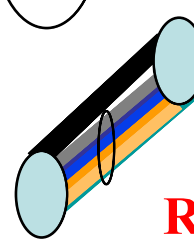
Sudoku







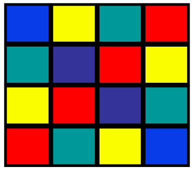
CONFLICT FREE LATIN ROUTER



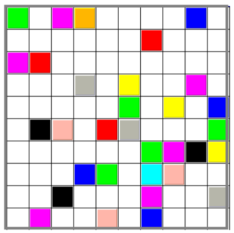
Routing in Fiber Optic Networks

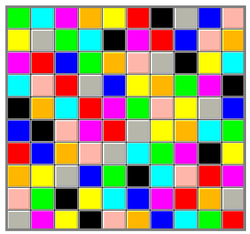
Many more applications…

Latin Square Completion



Latin Square (Order 4)

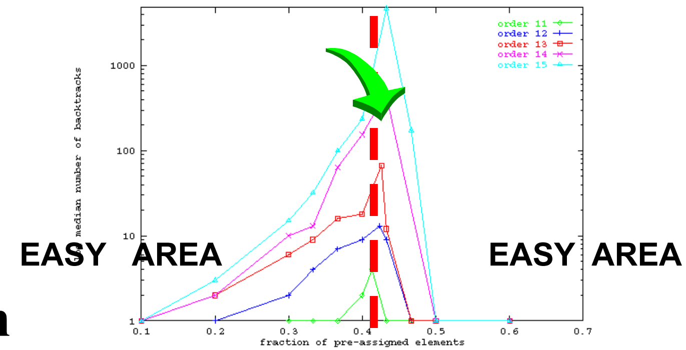




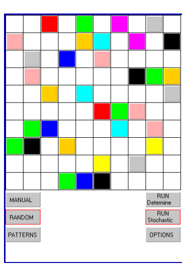
NP-Complete

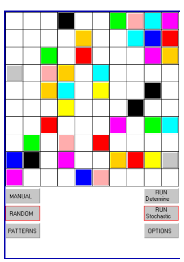
Better characterization beyond worst case

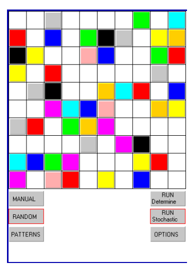
Critically constrained area



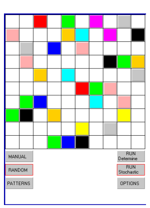
Complexity of Latin Square Completion

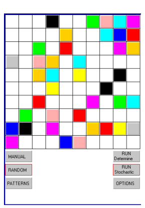


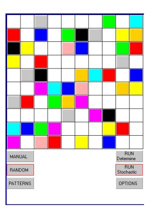




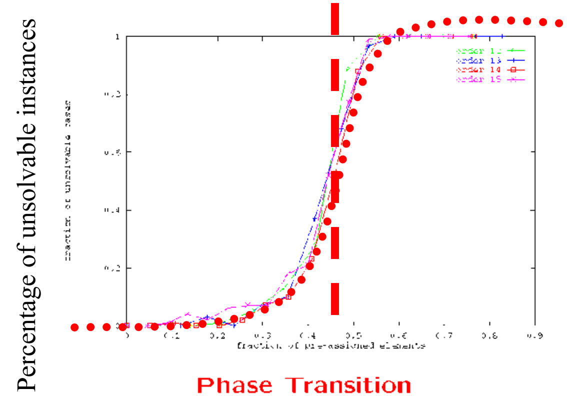
Time: 150 1820 165



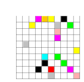


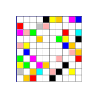


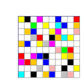
35% 42% 50%



Phase Transition







20% 42% 50%

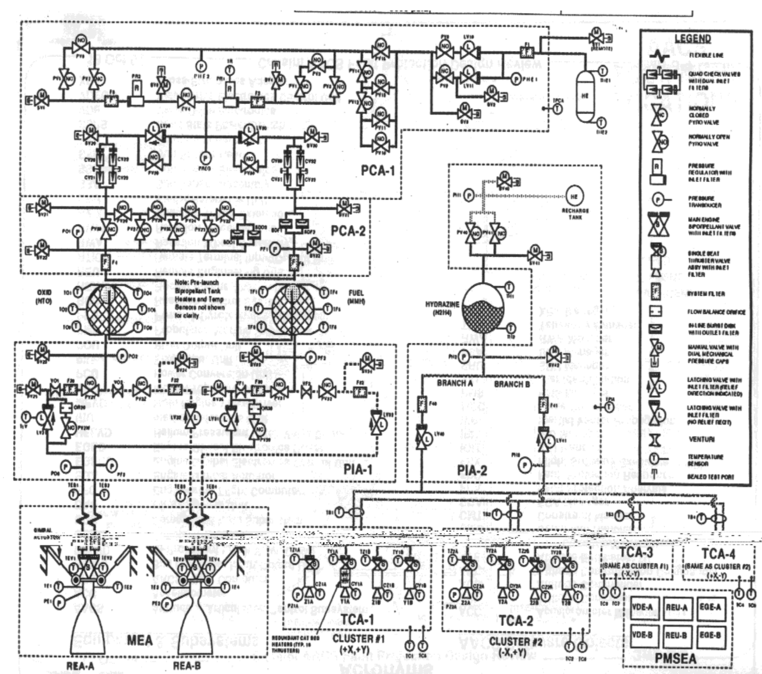
SAT Solvers in the Real World

NASA Deep Space One Spacecraft: Remote Agent

- Remote Agent (remote intelligent self-repair software) (RAX), developed at NASA and JPL, was the first artificial-intelligence control system to control a spacecraft without human supervision.
- Remote Agent successfully demonstrated the ability to plan onboard activities and correctly diagnose and respond to faults in spacecraft components through its built-in REPL environment

NASA Deep Space One: Remote Agent

- Autonomous diagnosis & repair "Remote Agent"
- Compiled systems schematic to 7,000 var SAT problem



Deep Space One

- a failed electronics unit
  - Remote Agent fixed by reactivating the unit.
- a failed sensor providing false information
  - Remote Agent recognized as unreliable and therefore correctly ignored.
- an altitude control thruster (a small engine for controlling the spacecraft's orientation) stuck in the "off" position
  - Remote Agent detected and compensated for by switching to a mode that did not rely on that thruster.

Significant progress in <u>Satisfiability Methods</u>

Software and hardware verification –
complete methods are critical - e.g. for
verifying the correctness of chip design, using
SAT encodings

Applications:
Hardware and
Software Verification
Planning,
Protocol Design, etc.

Going from 50 variable in, 200 constraints
to 1,000,000+ variables and 5,000,000+ constraints
in the last 20 years

Progress in Last 20 years

- *Significant progress since the 1990's.* How much?
- Problem size: **We went from 100 variables, 200 constraints (early 90's)**

  to 1,000,000+ variables and 5,000,000+ constraints in 20 years

- **Search space: from 10^30 to 10^300,000.**

  [Aside: "one can encode quite a bit in 1M variables."]

- **Is this just Moore's Law? It helped, but not much…**

  – 2x faster computers does *not* mean can solve 2x larger instances

  – search difficulty does *not* scale linearly with problem size!

  In fact, for O(2^n), 2x faster, how many more vars?

  handles 1 more variable!!

  Mainly algorithmic progress. Memory growth also key.

- **Tools: 50+ competitive SAT solvers available (e.g. Minisat solver)**
- **See http://www.satcompetition.org/**

Model Checking

Turing Award

2008 Turing Award Winners Announced

Posted by ScuttleMonkey **on Monday February 04, @05:30PM**
from the **nobel-of-computing-awards** dept.

The Association for Computing Machinery has announced the 2008 Turing Award Winners. Edmund M. Clarke, Allen Emerson, and Joseph Sifakis received the award for their work on an automated method for finding design errors in computer hardware and software.

> "Model Checking is a type of "formal verification" that analyzes the logic underlying a design, much as a mathematician uses a proof to determine that a theorem is correct. Far from hit or miss, Model Checking considers every possible state of a hardware or software design and determines if it is consistent with the designer's specifications. Clarke and Emerson originated the idea of Model Checking at Harvard in 1981. They developed a theoretical technique for determining whether an abstract model of a hardware or software design satisfies a formal specification, given as a formula in Temporal Logic, a notation for describing possible sequences of events. Moreover, when the system fails the specification, it could identify a counterexample to show the source of the problem. Numerous model checking systems have been implemented, such as Spin at Bell Labs."

Source: Slashdot

A "real world" example

From "SATLIB":

http://www.satlib.org/benchm.html

SAT-encoded bounded model checking instances (contributed by Ofer Shtrichman)

In Bounded Model Checking (BMC) [BCCZ99], a rather newly introduced problem in formal methods, the task is to check whether a given model M (typically a hardware design) satisfies a temporal property P in all paths with length less or equal to some bound k. The BMC problem can be efficiently reduced to a propositional satisfiability problem, and in fact if the property is in the form of an invariant (Invariants are the most common type of properties, and many other temporal properties can be reduced to their form. It has the form of 'it is always true that ... '.), it has a structure which is similar to many AI planning problems.

Bounded Model Checking instance:

The instance `bmc-ibm-6.cnf`, IBM LSU 1997:

```
p cnf 51639 368352
-1 7 0
-1 6 0
-1 5 0
-1 -4 0
-1 3 0
-1 2 0
-1 -8 0
-9 15 0
-9 14 0
-9 13 0
-9 -12 0
-9 11 0
-9 10 0
-9 -16 0
-17 23 0
-17 22 0
```

i.e. ((not x_1) or x_7)
and ((not x_1) or x_6)
and … *etc.*

10 pages later:

185 −9 0
185 −1 0
177 169 161 153 145 137 129 121 113 105 97
89 81 73 65 57 49 41
33 25 17 9 1 −185 0
186 −187 0
186 −188 0

…

(x_{177} or x_{169} or x_{161} or x_{153} …
or x_{17} or x_{9} or x_{1} or (not x_{185}))

clauses / constraints are getting more interesting…

4000 pages later:

10236 −10050 0
10236 −10051 0
10236 −10235 0
10008 10009 10010 10011 10012 10013 10014
10015 10016 10017 10018 10019 10020 10021
10022 10023 10024 10025 10026 10027 10028
10029 10030 10031 10032 10033 10034 10035
10036 10037 10086 10087 10088 10089 10090
10098 10099 10100 10101 10102 10103 10104
10105 10106 10107 10108 −55 −54 53 −52 −51 50
10047 10048 10049 10050 10051 10235 −10236 0
10237 −10008 0
10237 −10009 0
10237 −10010 0

…

!!!

a 59-cnf clause…

Finally, 15,000 pages later:

$-7\ 260\ 0$
$7\ -260\ 0$
$1072\ 1070\ 0$
$-15\ -14\ -13\ -12\ -11\ -10\ 0$
$-15\ -14\ -13\ -12\ -11\ 10\ 0$
$-15\ -14\ -13\ -12\ 11\ -10\ 0$
$-15\ -14\ -13\ -12\ 11\ 10\ 0$
$-7\ -6\ -5\ -4\ -3\ -2\ 0$
$-7\ -6\ -5\ -4\ -3\ 2\ 0$
$-7\ -6\ -5\ -4\ 3\ -2\ 0$
$-7\ -6\ -5\ -4\ 3\ 2\ 0$
$185\ 0$

Note that: $2^{50000} \approx 3.160699437 \cdot 10^{15051}$ *… !!!*

MiniSAT solver solves this instance in less than one minute.