Binary Decision Diagrams

[Randal Bryant'86]

Reduced Ordered Binary Decision Diagrams [Bryant'86]

- Compact represetation of boolean functions allowing effective manipulation (satifiability, validity,….)
- Compact representation of sets over finite universe allowing effective manipulations.

Use of ROBDDs

- Comibatorial circuits
- Sequential circuits
- Automata
- **Combinatorial problems**
- **Temporal logic model checking**
- Program analysis
- …..

Boolean Functions

Boolean functions: $\mathbb{B} = \{0, 1\}$,

$$f : \mathbb{B} \times \cdots \times \mathbb{B} \to \mathbb{B}$$

Boolean expressions:

$$t ::= x \mid 0 \mid 1 \mid \neg t \mid t \wedge t \mid t \vee t \mid t \Rightarrow t \mid t \Leftrightarrow t$$

Truth assignments: ρ,

$$[v_1/x_1, v_2/x_2, \ldots, v_n/x_n]$$

Satisfiable: Exists ρ such that $t[\rho] = 1$

Tautology: Forall ρ, $t[\rho] = 1$

Truth Tables

	$\neg$
0	1
1	0

$\wedge$	0	1
0	0	0
1	0	1

$\vee$	0	1
0	0	1
1	1	1

$\Rightarrow$	0	1
0	1	1
1	0	1

$\Leftrightarrow$	0	1
0	1	0
1	0	1

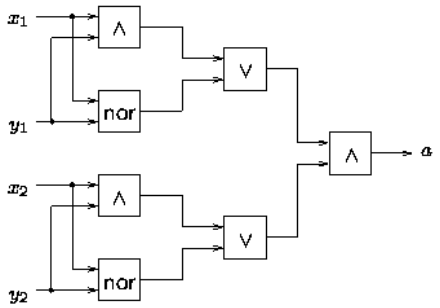
xyz	$x \rightarrow y, z$
000	0
001	1
010	0
011	1
100	0
101	0
110	1
111	1

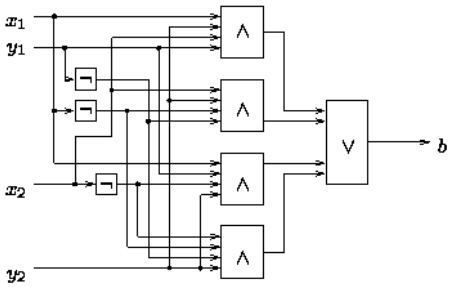
$x_1 \cdots x_n$	$f(x_1, \ldots, x_n)$
$0 \cdots 0$	1
$0 \cdots 1$	0
$\vdots$	$\vdots$
$1 \cdots 1$	0

2^n entries in table!

5

Combinatorial Circuits



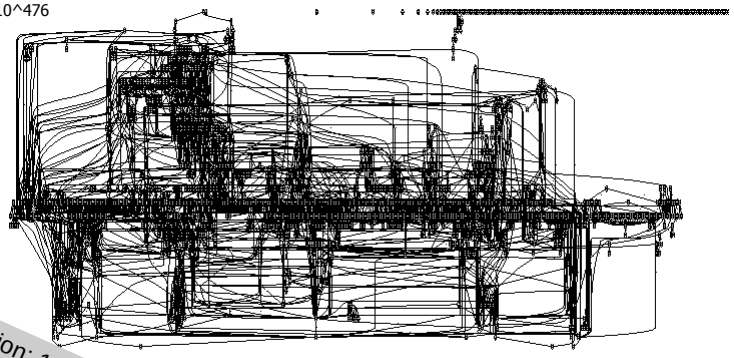


Are they two circuits equivalent?

Control Programs
A Train Simulator, visualSTATE (VVS)

1421 machines
11102 transitions
2981 inputs
2667 outputs
3204 local states
Declare state sp.: 10^476

BUGS ?



"Ideal" presentation: 1 bit/state will clearly NOT work!

"Good" Representations of Boolean Functions

Always perfect representations are hopeless

Normalforms

- Disjunctive NF
- Conjunctive NF
- If-then-else NF
- …….

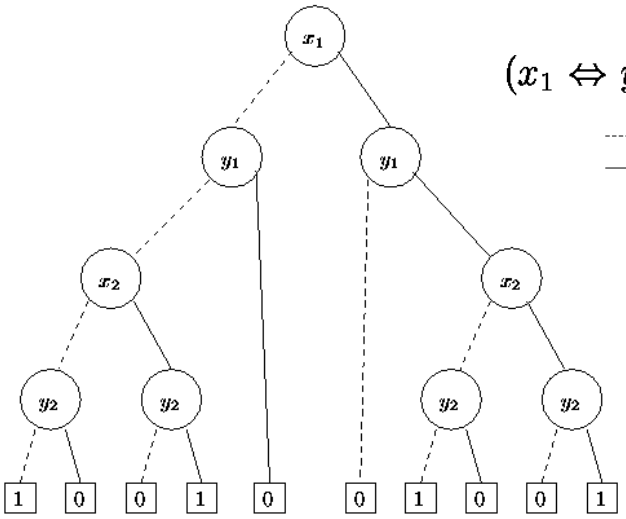
THEOREM (Cook's theorem)

Satisfiability of Boolean expressions is NP-complete

Compact representations are
- **compact** and
- **efficient**

on **real-life** examples

Binary Decision *Trees*



$(x_1 \Leftrightarrow y_1) \wedge (x_2 \Leftrightarrow y_2)$

Variable is set to 0
Variable is set to 1

Each path determines a partial (set of) truth assignments.
Result of the boolean expression under the given assigment found in value of terminal.

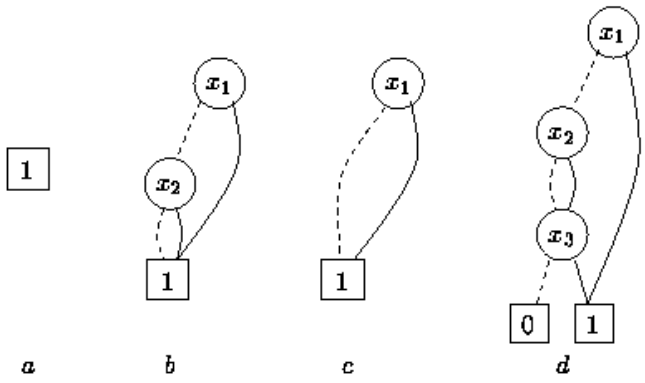
Binary Decision *Diagrams*
allows NODES to be shared

Equivalence ~ on nodes:

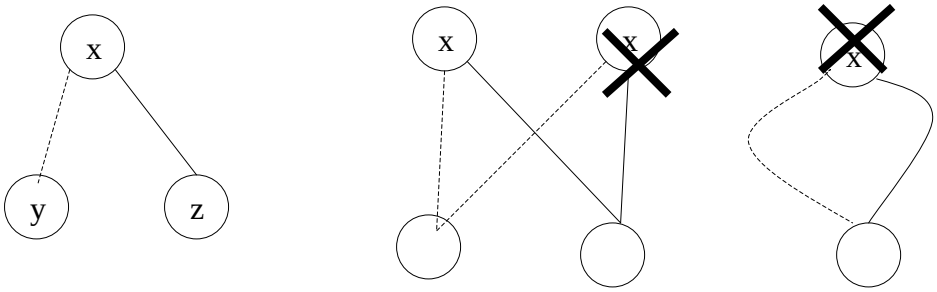
n ~ *m* **iff**
either both *n* and *m* are terminals
and have the same value
or both are non-terminals with var(*n*) = var(*m*) **and**
1. *n'* ~ *m'* **when** *n* -0-> *n'* , *m*-0->*m'* , **and**
2. *n'* ~ *m'* **when** *n* -1-> *n'* , *m*-1->*m'*

Have you seen this somewhere before?

Orderedness & Redundant *TESTS*

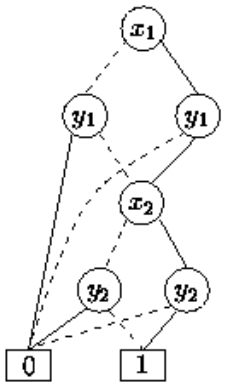


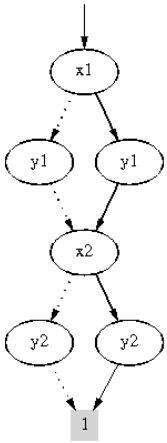
Orderedness & Reducedness



x<y x<z

Reduced Ordered Binary Decision Diagrams





Iben
Edges to 0 implicit

ROBDDs formally

A *Binary Decision Diagram* is a rooted, directed, acyclic graph (V, E). V contains (up to) two *terminal* vertices, $0, 1 \in V$. $v \in V \setminus \{0,1\}$ are *non-terminal* and has attributes *var*(v), and *low*(v), *high*$(v) \in V$.

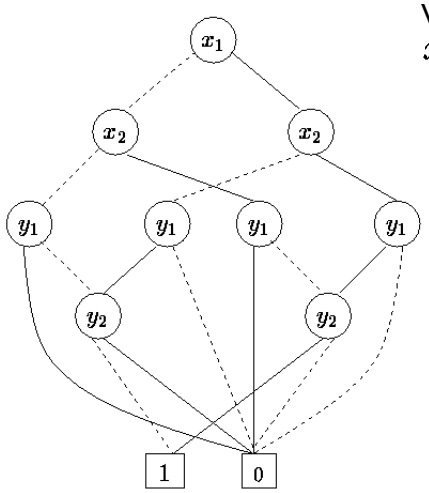
A BDD is *ordered* if on all paths from the root the variables respect a given total order.

A BDD is *reduced* if for all non-terminal vertices u, v,

1) *low*$(u) \neq$ *high*(u)
2) *low*$(u) =$ *low*(v), *high*$(u) =$ *high*(v), *var*$(u) =$ *var*(v) implies $u = v$

Ordering *does* matter!

Variable ordering
$x_1 < x_2 < y_1 < y_2$



Canonicity of ROBDDs

$t_0 = 0$
$t_1 = 1$
$t_u = x \to t_h, t_l,$ if u is a node (x, l, h)

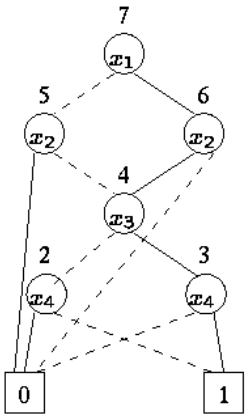
Lemma 1 (Canonicity lemma) *For any function $f : \mathbb{B}^n \to \mathbb{B}$ there is exactly one ROBDD b with variables $x_1 < x_2 < \ldots < x_n$ such that*

$$t_b[v_1/x_1, \ldots, v_n/x_n] = f(v_1, \ldots, v_n)$$

for all $(v_1, \ldots, v_n) \in \mathbb{B}^n$.

Consequences:
b is a tautology, if and only if, $b = \boxed{1}$
b is satisfiable, if and only if, $b \neq \boxed{0}$

Array implementation



$root = 7$

	var	*low*	*high*
0	?	?	?
1	?	?	?
2	4	1	0
3	4	0	1
4	3	2	3
5	2	4	0
6	2	0	4
7	1	5	6

b

Makenode and Hashing

Makenode ensures reducedness using a hash table

$$H : (i, l, h) \mapsto u$$

Makenode(H, *max*, b, i, l, h)
1: **if** $l = h$ **then return** l
2: **else if** *member*(H, i, l, h) **then return** *lookup*(H, i, l, h)
3: **else** $max \leftarrow max + 1$
4: $b.var(max) \leftarrow i$
5: $b.low(max) \leftarrow l$
6: $b.high(max) \leftarrow h$
7: *insert*(H, i, l, h, *max*)
8: **return** *max*

BUILD

Build: Maps a Boolean expression, t, into an ROBDD.

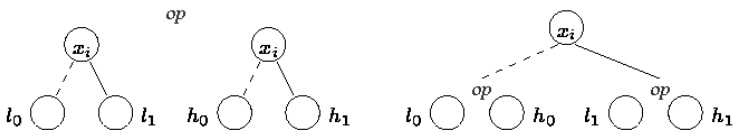
```
Build(t)
3:   function build'(t, i) =
4:       if i > n then if t is false then return 0 else return 1
5:       else l ← build'(t[0/x_i], i + 1)
6:            h ← build'(t[1/x_i], i + 1)
7:            return makenode(H, max, b, i, l, h)
8:   end build'
9:
1:   H ← emptytable   max ← 1
10:  b.root ← build'(t, 1)
11:  return b
```

Run time?

Boolean operations on ROBDDs

Ordering: $x_1 < \cdots < x_n$

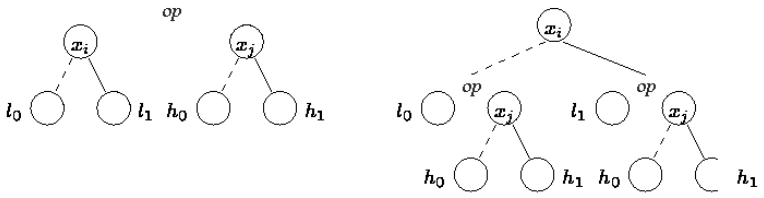
$$(x_i \to l_1, l_0)\ op\ (x_i \to h_1, h_0) = x_i \to (l_1\ op\ h_1), (l_0\ op\ h_0)$$



Boolean operations on ROBDDs

If $x_i < x_j$:

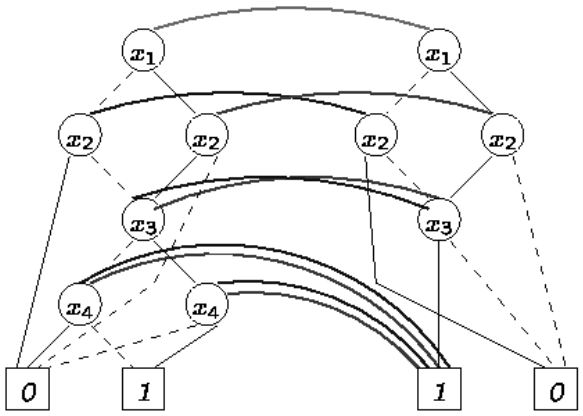
$$x_i \to l_1, l_0) \ op \ (x_j \to h_1, h_0) =$$
$$x_i \to (l_1 \ op \ (x_j \to h_1, h_0)), (l_0 \ op \ (x_j \to h_1, h_0))$$



APPLY operation

```
Apply(op, b1, b2)
4:   function app(u1, u2) =
6:            if u1 ∈ {0,1} and u2 ∈ {0,1} then res ← op(u1, u2)
7:            else if u1 ∈ {0,1} and u2 ≥ 2 then
8:                 res ← makenode(var(u2), app(u1, low(u2)), app(u1, high(u2))
9:            else if u1 ≥ 2 and u2 ∈ {0,1} then
10:                res ← makenode(var(u1), app(low(u1), u2), app(high(u1), u2))
11:           else if var(u1) = var(u2) then
12:                res ← makenode(var(u1), app(low(u1), low(u2)),
                                           app(high(u1), high(u2)))
13:           else if var(u1) < var(u2) then
14:                res ← makenode(var(u1), app(low(u1), u2), app(high(u1), u2))
15:           else (* var(u1) > var(u2) *)
16:                res ← makenode(var(u2), app(u1, low(u2)), app(u1, high(u2)))
18:           return res
20:
21:  b.root ← app(b1.root, b2.root)
22:  return b
```

APPLY example



APPLY with dynamic programming

```
Apply(op, b1, b2)
4:   function app(u1, u2) =
5:        if G(u1, u2) ≠ empty then return G(u1, u2)
6:        else if u1 ∈ {0,1} and u2 ∈ {0,1} then res ← op(u1, u2)
7:             else if u1 ∈ {0,1} and u2 ≥ 2 then
8:                  res ← makenode(var(u2), app(u1, low(u2)), app(u1, high(u2))
9:             else if u1 ≥ 2 and u2 ∈ {0,1} then
10:                 res ← makenode(var(u1), app(low(u1), u2), app(high(u1), u2))
11:            else if var(u1) = var(u2) then
12:                 res ← makenode(var(u1), app(low(u1), low(u2)),
                                            app(high(u1), high(u2)))
13:            else if var(u1) < var(u2) then
14:                 res ← makenode(var(u1), app(low(u1), u2), app(high(u1), u2))
15:            else (* var(u1) > var(u2) *)
16:                 res ← makenode(var(u2), app(u1, low(u2)), app(u1, high(u2)))
17:            G(u1, u2) ← res
18:            return res
20:
2:   forall i ≤ max(b1), j ≤ max(b2) : G(i, j) ← empty
21:  b.root ← app(b1.root, b2.root)
22:  return b
```

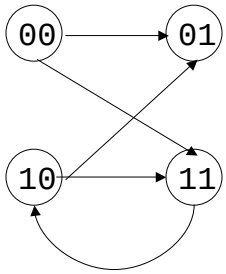
Other operations

Restrict	$b[v/x]$
Size (satcount)	$\mathit{size}(b) = \lvert\{\rho \mid b[\rho] = 1\}\rvert$
Anysat	$\mathit{anysat}(b) = \rho$, for some ρ with $b[\rho] = 1$
Allsat	$\mathit{allsat}(b) = \{\rho \mid b[\rho] = 1\}$
Compose	$\mathit{compose}(b, x, b') = b[x/b']$
Existential quantification	$\exists x.b = b[x/0] \vee b[x/1]$

ROBDDs and Verification

[…,McMillan'90,…..,VVS]

ROBDD encoding of transition system

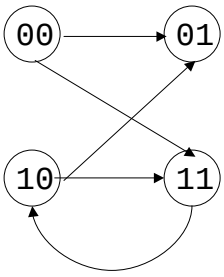


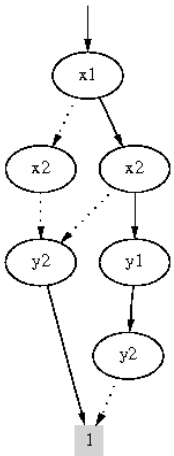
Encoding of states using binary variables (here `x1` and `x2`).

Encoding of transition relation using source and target variables (here `x1, x2, y1`, and `y2`)

```
Trans(x1,x2,y1,y2):=
     !x1 & !x2 & !y1 & y2
   + !x1 & !x2 &  y1 & y2
   +  x1 & !x2 & !y1 & y2
   +  x1 & !x2 &  y1 & y2
   +  x1 &  x2 &  y1 & !y2;
```


ROBDD representation (cont.)

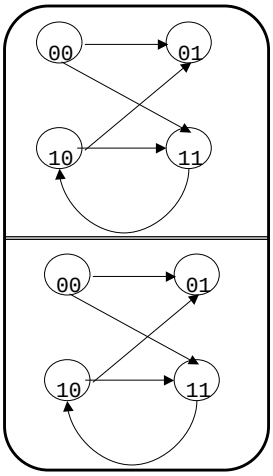





```
Trans(x1,x2,y1,y2):=
     !x1 & !x2 & !y1 & y2
   + !x1 & !x2 &  y1 & y2
   +  x1 & !x2 & !y1 & y2
   +  x1 & !x2 &  y1 & y2
   +  x1 &  x2 &  y1 & !y2;
```

ROBDD for parallel composition

ATrans(**x,y**)



BTrans(**u,v**)

Asynchronous composition

```
Trans(x,y,u,v) =
      (ATrans(x,y) & v=u)
    + (BTrans(u,v) & y=x)
```

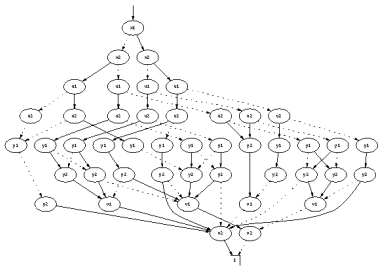
Synchronous composition

```
Trans(x,y,u,v) =
    (ATrans(x,y) & BTrans(u,v))
```

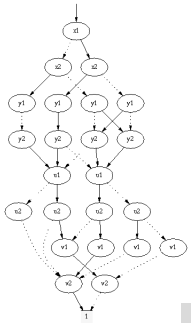
Which ordering to choose?

Ordering?

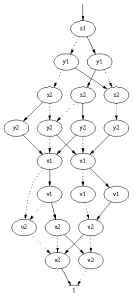
45 nodes
x1,x2,u1,u2, y1,y2 ,v1,v2



23 nodes
x1,x2,y1,y2,u1,u2,v1,v2



20 nodes
x1,y1,x2,y2,u1,v1,u2,v2

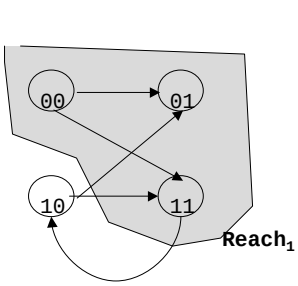


Polynomial size BDDs guaranteed in size of argument BDDs [Enders,Filkorn, Taubner'91]

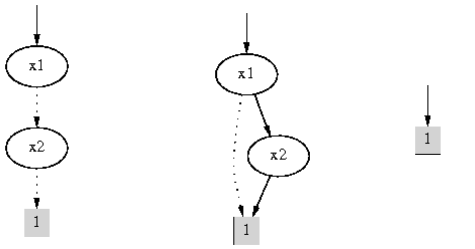
Reachable States

```
Reach(x) := Init(x);
REPEAT
   Old(x) := Reach(x);
   New(y) := Exists x.(Reach(x) & Trans(x,y));
   Reach(x) := Old(x) + New(x)
UNTIL Old(x) = Reach(x)
```

Relational Product: May be constructed without building intermediate (often large) &-BDD.

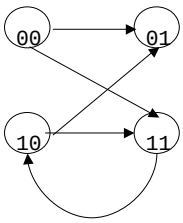


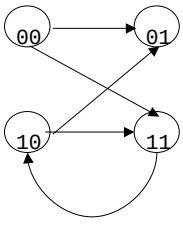
Reach0 **Reach1** **Reach2**



Bisimulation

vars **x (y)**

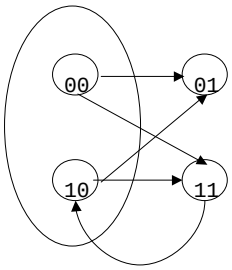




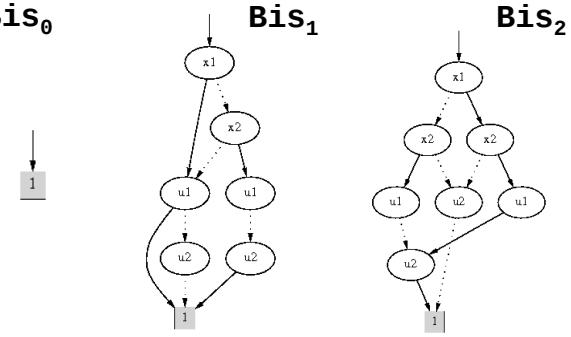
vars **u (v)**

```
Bis(x,u):= 1;
REPEAT
   Old(x,u) := Bis(x,u);
   Bis(x,u) :=
      Forall y. Trans(x,y) =>
         (Exists v. Trans(u,v) & Bis(y,v))
      &
      Forall v. Trans(u,v) =>
         (Exists y. Trans(x,y) & Bis(y,v));
UNTIL Bis(x,u)=Old(x,u)
```

Bisimulation (cont.)



$\mathbf{Bis_0}$ $\mathbf{Bis_1}$ $\mathbf{Bis_2}$



3 equivalence classes
= 6 pairs in final bisimulation

Symbolic Modelchecking CTL

f ::= p | ~*f* | *f* & *g* | **EX** *f* | **E**[*f* **U***g*] | **EG** *f*

CHECK:

INPUT:

- A CTL-formula *f*
- BDD representation of transition relation `T(x,y)`.

OUTPUT

- BDD `S(x)` of set of states satisfying *f* .

CHECK defined on structure of *f*

CHECK

- CHECK(p): BDD representing states satisfying atomic proposition p.
- CHECK(**EX** *f*) = CHECKEX(CHECK(*f*))
- CHECK(**E**[*f* **U** *g*]) = CHECKEU(CHECK(*f*),CHECK(*g*))
- CHECK(**EG** *f*) = CHECKEG(CHECK(*f*))

Boolean connectives are handled using standard BDD operations

CHECKEX

```
CHECKEX( S(x) ) =
      Exists y. (T(x,y) & S(y))
```

CHECKEU

CHECKEU(Sf(**x**),Sg(**x**)) is the least fixpoint to the following equation:

$$X(\mathbf{x}) = Sg(\mathbf{x}) + (Sf(\mathbf{x}) \;\&\; CHECKEX(X(\mathbf{x})))$$

CHECKEG

CHECKEU(Sf(**x**)) is the greatest fixpoint to the following equation:

$$X(\mathbf{x}) = Sf(\mathbf{x}) \;\&\; CHECKEX(X(\mathbf{x}))$$

Partitioned Transition Relation

Relational Product

LARGE

```
Exists yv. (T(xu,yv) & S(yv))
```

Asynchronous

```
T(xy,uv) =
      (ATrans(x,y) & v=u)
    + (BTrans(u,v) & y=x)
```

Synchronous

```
T(xy,uv) =
    ATrans(x,y)
  & BTrans(u,v)
```

```
Exists yv. AT(xu,yv) & S(yv)
+
Exists yv. BT(xu,yv) & S(yv)
```

```
Exists yv.
  Atrans(x,y)
& Btrans(u,v) & S(yv)
```


BAAN visualSTATE (Beologic) CIT project VVS (w DTU)

Beologic's Products: salesPLUS visualSTATE

•Embedded Systems
•Simple Model
•Verification of Std. Checks
•Explicit Representation
*(**STATEEXPLOSION**)*
•Code Generation

1980-95: *Independent division of B&0*
1995- : *Independent company*
B&O, 2M Invest,
Danish Municipal Pension Ins. Fund

Customers:

ABB
B&O
Daimler-Benz
Ericson DIAX
ESA/ESTEC
FORD
Grundfos
LEGO
PBS
Siemens ……. (approx. 200)

Verification Problems:
- 1.400 components
- 10^{400} states

Our techniques has reduced verification by an order of magnitude (from 14 days to 6 sec)

Experimental Breakthroughs

patent pending

System	Mach.	State Space		Checks	Visual ST	St-of-Art		ComBack	
		Declared	Reach			Sec	MB	Sec	MB
VCR	7	10^5	1279	50	<1	<1	6	<1	7
JVC	8	10^4	352	22	<1	<1	6	<1	6
HI-FI	9	10^7	1416384	120	1200	1.0	6	3.9	6
Motor	12	10^7	34560	123	32	<1	6	2,0	
AVS	12	10^7	1438416	173	3780	6.7	6	5.7	6
Video	13	10^8	1219440	122	---	1.1	6	1.5	6
Car	20	10^11	9.2 10^9	83	---	3.8	9	1.8	6
N6	14	10^10	6399552	443	---	32.3	7	218	6
N5	25	10^12	5.0 10^10	269	---	56.2	7	9.1	6
N4	23	10^13	3.7 10^8	132	---	622	7	6.3	6
Train1	**373**	**10^136**	**---**	**1335**	**---**	**---**	**---**	**25.9**	**6**
Train2	**1421**	**10^476**	**---**	**4708**	**---**	**---**	**---**	**739**	**11**

Machine: 166 MHz Pentium PC with 32 MB RAM

---: Out of memory, or did not terminate after 3 hours.

Experimental Breakthroughs

patent pending

System	Mach.	State Space		Checks	Visual ST	St-of-Art		ComBack	
		Declared	Reach			Sec	MB	Sec	MB
VCR	7	10^5	1279	50				<1	7
JVC	8	10^4	352					<1	6
HI-FI	9	10^7	1416384					3.9	6
Motor	12	10^7					6	2,0	
AVS	12	10^7					6	5.7	6
Video	13	10^				1.1	6	1.5	6
Car	20				---	3.8	9	1.8	6
N6	1			443	---	32.3	7	218	6
N5				269	---	56.2	7	9.1	6
N4			10^8	132	---	622	7	6.3	6
Train1			**---**	**1335**	**---**	**---**	**---**	**25.9**	**6**
Train2	**14**		**---**	**4708**	**---**	**---**	**---**	**739**	**11**

Our technique have reduced verification time by several orders of magnitude (eg. From 14 days to 6 sec)

Machine: 166 MHz Penti PC with 32 MB RAM

---: Out of memory, or did not terminate after 3 hours.