Finite Model Search in First-Order Logic

William McCune
Mathematics and Computer Science Division
Argonne National Laboratory

October 2001

Proofs and Countermodels

- Ordinary first-order theorem proving:
  - search for a proof of a conjecture,
  - show that a statement is unsatisfiable.
- We'll look at the other side of the question:
  - search for a counterexample to a conjecture,
  - show that a statement has a model.

Approaches

- Saturation with a complete inference system.
  - for example, unfailing Knuth-Bendix completion
  - if it terminates without a proof, the statement is satisfiable
  - but we do not necessarily get a model
- Search for finite models.
  - iterate through domain sizes: 2,3,4,...
  - each domain size is decidable by reduction to a ground problem.
  - it is practical for small statements and small domains
  - example programs: FINDER, MGTP, MACE, FALCON, STORK, SEM.
  - also, there are *local search* (incomplete) methods

Noncommutative Group Example

Input

```
op(400, xfx, *).
op(350, yf, ~).

list(usable).

e * x = x.
x~ * x = e.
(x * y) * z = x * (y * z).

a * b != b * a.

end_of_list.
```

Noncommutative Group Example

Input

```
op(400, xfx, *).
op(350, yf, ~).

list(usable).

e * x = x.
x~ * x = e.
(x * y) * z = x * (y * z).

a * b != b * a.

end_of_list.
```

Output

```
Model #1 at 0.32 seconds:

 * | 0 1 2 3 4 5
 --+------------
 0 | 0 1 2 3 4 5
 1 | 1 0 3 2 5 4
 2 | 2 4 0 5 1 3
 3 | 3 5 1 4 0 2
 4 | 4 2 5 0 3 1
 5 | 5 3 4 1 2 0

 ~   0 1 2 3 4 5
 ---------------
     0 1 2 4 3 5

e: 0  a: 1  b: 2
```

Quantum Logic

Lattice Axioms

$$x \wedge y = y \wedge x \qquad x \vee y = y \vee x$$
$$(x \wedge y) \wedge z = x \wedge (y \wedge z) \qquad (x \vee y) \vee z = x \vee (y \vee z)$$
$$(z \vee y) \wedge x = x \qquad (x \wedge y) \vee x = x$$

Add these to get an Ortholattice

$$x \vee c(x) = 1 \qquad x \wedge c(x) = 0$$
$$c(c(x)) = x \qquad c(x \vee y) = c(x) \wedge c(y)$$

Weak Orthomodular law

$$(c(x) \wedge (x \vee y)) \vee (c(y) \vee (x \wedge y)) = 1$$

Quantum Logic

Does the following hold in all weakly orthomodular lattices? [Norm Megill].

$$x \wedge (y \vee (x \wedge (c(x) \vee (x \wedge y)))) = x \wedge (c(x) \vee (x \wedge y)) \qquad \text{(*3-M68)}$$

Quantum Logic

Does the following hold in all weakly orthomodular lattices? [Norm Megill].

$$x \wedge (y \vee (x \wedge (c(x) \vee (x \wedge y)))) = x \wedge (c(x) \vee (x \wedge y)) \qquad \text{(*3-M68)}$$

No! [SEM].

Quantum Logic

m	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	1	2	6	1	17	6	9	10	9	10	9	15	2	15	15	1	17	17	17
3	3	1	6	3	1	1	6	1	6	1	6	1	3	3	3	6	1	1	1	1
4	4	1	1	1	4	16	1	4	4	1	1	4	4	1	1	1	16	1	4	16
5	5	1	17	1	16	5	1	5	16	17	1	19	16	17	1	1	16	17	19	19
6	6	1	6	6	1	1	6	1	6	1	6	1	6	6	6	6	1	1	1	1
7	7	1	9	1	4	5	1	7	4	9	1	11	4	9	1	1	16	17	18	19
8	8	1	10	6	4	16	6	4	8	1	10	4	8	10	10	10	16	1	4	16
9	9	1	9	1	1	17	1	9	1	9	1	9	1	9	1	1	1	17	17	17
10	10	1	10	6	1	1	6	1	10	1	10	1	10	10	10	10	1	1	1	1
11	11	1	9	1	4	19	1	11	4	9	1	11	4	9	1	1	16	17	18	19
12	12	1	15	3	4	16	6	4	8	1	10	4	12	14	14	15	16	1	4	16
13	13	1	2	3	1	17	6	9	10	9	10	9	14	13	14	15	1	17	17	17
14	14	1	15	3	1	1	6	1	10	1	10	1	14	14	14	15	1	1	1	1
15	15	1	15	6	1	1	6	1	10	1	10	1	15	15	15	15	1	1	1	1
16	16	1	1	1	16	16	1	16	16	1	1	16	16	1	1	1	16	1	16	16
17	17	1	17	1	1	17	1	17	1	17	1	17	1	17	1	1	1	17	17	17
18	18	1	17	1	4	19	1	18	4	17	1	18	4	17	1	1	16	17	18	19
19	19	1	17	1	16	19	1	19	16	17	1	19	16	17	1	1	16	17	19	19

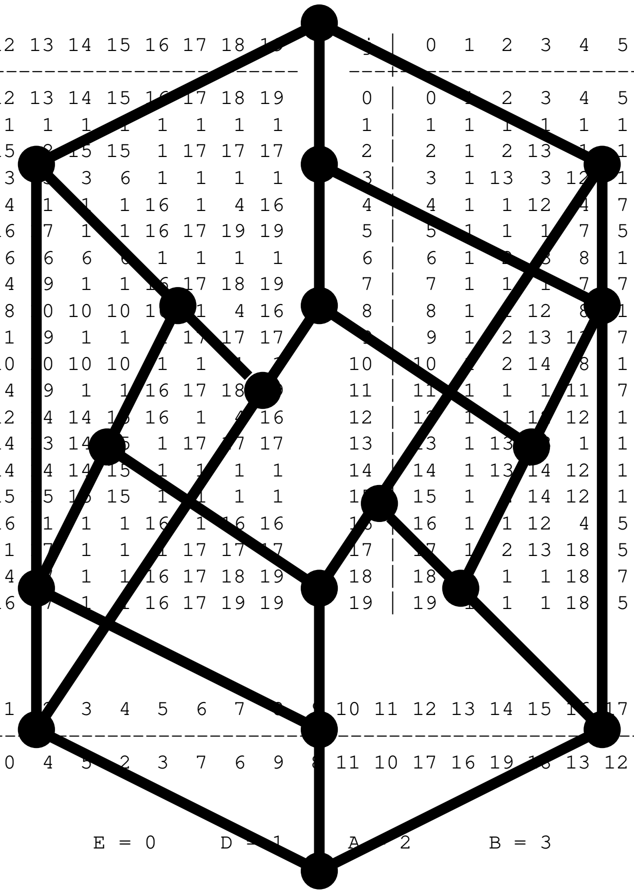
j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	1	2	13	1	1	2	1	1	2	2	1	1	13	13	2	1	2	1	1
3	3	1	13	3	12	1	3	1	12	13	14	1	12	13	14	14	12	13	1	1
4	4	1	1	12	4	7	8	7	8	11	8	11	12	1	12	12	4	18	18	18
5	5	1	1	1	7	5	1	7	1	7	1	7	1	1	1	1	5	5	7	5
6	6	1	2	3	8	1	6	1	8	2	10	1	12	13	14	15	8	2	1	1
7	7	1	1	1	7	7	1	7	1	7	1	7	1	1	1	1	7	7	7	7
8	8	1	1	12	8	1	8	1	8	1	8	1	12	1	12	12	8	1	1	1
9	9	1	2	13	11	7	2	7	1	9	2	11	1	13	13	2	11	9	11	11
10	10	1	2	14	8	1	10	1	8	2	10	1	12	13	14	15	8	2	1	1
11	11	1	1	1	11	7	1	7	1	11	1	11	1	1	1	1	11	11	11	11
12	12	1	1	12	12	1	12	1	12	1	12	1	12	1	12	12	12	1	1	1
13	13	1	13	13	1	1	13	1	1	13	13	1	1	13	13	13	1	13	1	1
14	14	1	13	14	12	1	14	1	12	13	14	1	12	13	14	14	12	13	1	1
15	15	1	2	14	12	1	15	1	12	2	15	1	12	13	14	15	12	2	1	1
16	16	1	1	12	4	5	8	7	8	11	8	11	12	1	12	12	16	19	18	19
17	17	1	2	13	18	5	2	7	1	9	2	11	1	13	13	2	19	17	18	19
18	18	1	1	1	18	7	1	7	1	11	1	11	1	1	1	1	18	18	18	18
19	19	1	1	1	18	5	1	7	1	11	1	11	1	1	1	1	19	19	18	19

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
c	1	0	4	5	2	3	7	6	9	8	11	10	17	16	19	18	13	12	15	14

E = 0   D = 1   A = 2   B = 3

An Weakly Orthomodular Lattice Violating (*3-M68)

Quantum Logic



m	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	1	2	6	1	17	6	9	10	9	10	9	15	[illegible]	15	15	1	17	17	17
3	3	1	6	3	1	1	6	1	6	1	6	1	3	[illegible]	3	6	1	1	1	1
4	4	1	1	1	4	16	1	4	4	1	1	4	4	[illegible]	1	1	16	1	4	16
5	5	1	17	1	16	5	1	5	16	17	1	19	16	[illegible]	1	1	16	17	19	19
6	6	1	6	6	1	1	6	1	6	1	6	1	6	[illegible]	6	6	1	1	1	1
7	7	1	9	1	4	5	1	7	4	9	1	11	4	[illegible]	1	1	16	17	18	19
8	8	1	10	6	4	16	6	4	8	1	10	4	8	[illegible]	10	10	[illegible]	[illegible]	4	16
9	9	1	9	1	1	17	1	9	1	9	1	9	1	[illegible]	1	1	[illegible]	17	17	17
10	10	1	10	6	1	1	6	1	10	1	10	1	10	[illegible]	10	10	[illegible]	1	[illegible]	[illegible]
11	11	1	9	1	4	19	1	11	4	9	1	11	4	[illegible]	1	1	16	17	18	[illegible]
12	12	1	15	3	4	16	6	4	8	1	10	4	12	[illegible]	14	[illegible]	16	1	[illegible]	16
13	13	1	2	3	1	17	6	9	10	9	10	9	14	[illegible]	14	[illegible]	1	17	[illegible]	17
14	14	1	15	3	1	1	6	1	10	1	10	1	14	[illegible]	14	15	1	1	1	1
15	15	1	15	6	1	1	6	1	10	1	10	1	15	[illegible]	[illegible]	15	1	1	1	1
16	16	1	1	1	16	16	1	16	16	1	1	16	16	[illegible]	1	1	16	1	16	16
17	17	1	17	1	1	17	1	17	1	17	1	17	1	[illegible]	1	1	[illegible]	17	17	[illegible]
18	18	1	17	1	4	19	1	18	4	17	1	18	4	[illegible]	1	1	16	17	18	19
19	19	1	17	1	16	19	1	19	16	17	1	19	16	[illegible]	1	1	16	17	19	19

[illegible]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	[illegible]	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	1	2	13	[illegible]	1	2	1	1	2	2	1	1	13	13	2	1	2	1	1
3	3	1	13	3	[illegible]	1	3	1	12	13	14	1	12	13	14	14	12	13	1	1
4	4	1	1	12	[illegible]	7	8	7	8	11	8	11	12	1	12	12	4	18	18	18
5	5	1	1	1	7	5	1	7	1	7	1	7	1	1	1	1	5	5	7	5
6	6	1	[illegible]	[illegible]	8	1	6	1	8	2	10	1	12	13	14	15	8	2	1	1
7	7	1	1	1	7	7	1	7	1	7	1	7	1	1	1	1	7	7	7	7
8	8	1	[illegible]	12	[illegible]	1	8	1	8	1	8	1	12	1	12	12	8	1	1	1
[illegible]	9	1	2	13	[illegible]	7	2	7	1	9	2	11	1	13	13	2	11	9	11	11
10	10	[illegible]	2	14	[illegible]	1	10	1	8	2	10	1	12	13	14	15	8	2	1	1
11	11	1	1	1	11	7	1	7	1	11	1	11	1	1	1	1	11	11	11	11
12	[illegible]	1	1	[illegible]	12	1	12	1	12	1	12	1	12	1	12	12	12	1	1	1
13	[illegible]	1	13	[illegible]	1	1	13	1	1	13	13	1	1	13	13	13	1	13	1	1
14	14	1	13	14	12	1	14	1	12	13	14	1	12	13	14	14	12	13	1	1
[illegible]	15	1	[illegible]	14	12	1	15	1	12	2	15	1	12	13	14	15	12	2	1	1
[illegible]	16	1	[illegible]	12	4	5	8	7	8	11	8	11	12	1	12	12	16	19	18	19
17	[illegible]	[illegible]	2	13	18	5	2	7	1	9	2	11	1	13	13	2	19	17	18	19
18	18	[illegible]	1	1	18	7	1	7	1	11	1	11	1	1	1	1	18	18	18	18
19	19	[illegible]	1	1	18	5	1	7	1	11	1	11	1	1	1	1	19	19	18	19

	0	1	[illegible]	3	4	5	6	7	[illegible]	[illegible]	10	11	12	13	14	15	16	17	18	19
c	1	0	4	5	2	3	7	6	9	8	11	10	17	16	19	[illegible]	13	12	15	14

E = 0 D = 1 A = 2 B = 3

An Weakly Orthomodular Lattice Violating (*3-M68)

Single Axioms for Boolean Algebra

- Is this a single axiom for Boolean Algebra in terms of OR and NOT?

$$\neg(\neg(x \vee \neg(\neg(y \vee x) \vee x)) \vee \neg(y \vee \neg(z \vee x))) = y.$$

Single Axioms for Boolean Algebra

- Is this a single axiom for Boolean Algebra in terms of OR and NOT?

$$\neg(\neg(x \vee \neg(\neg(y \vee x) \vee x)) \vee \neg(y \vee \neg(z \vee x))) = y.$$

- No. SEM finds a countermodel — a model in which $\exists A (A \vee A) \neq A$.

```
or | 0 1 2 3     not | 0 1 2 3
---+--------        ---+--------      A = 0
 0 | 1 0 3 2           | 0 1 2 3
 1 | 0 0 0 0
 2 | 3 0 3 0
 3 | 2 0 0 2
```

- This job is too big for MACE.

The Zebra Puzzle

- There are five houses in a row: 0,1,2,3,4.
  Each has a color, a person, a pet, a drink, and a food.

- John lives in the Red house. Bill has a Dog. Coffee is in the Green house. Mary drinks Tea. The cat eats pizza. Rice is in the Brown house. Water is in house 2. Sue lives in house 0. Wine goes with Pasta. Clara eats Noodles. House 1 is Blue.

- The Brown house is just to the left of the Green house. The Horse is next to the house with Rice. The Mouse is next to the house with Beans.

How many solutions are there?

Zebra Puzzle: Representation

- Domain of 5 objects, one for each house. Solutions will look like

$$1 = Red = John = Dog = Coffee = Pizza.$$

- Basic properties: People are different, Colors are different, etc.

```
John!=Bill. John!=Mary. John!=Sue. John!=Clara. Bill!=Mary.
Bill!=Sue. Bill!=Clara. Mary!=Sue. Mary!=Clara. Sue!=Clara.
etc.
```

- Simple Clues

```
John=Red. Bill=Dog. Coffee=Green. Mary=Tea. Pizza=Cat.
Rice=Brown. Water=2. Sue=0. Pasta=Wine. Clara=Noodles. Blue=1.
```

Zebra Puzzle: Remaining clues

- Define successor and neighbor relations

```
all x y (successor(x,y) <-> (x=0 & y=1) |
                             (x=1 & y=2) |
                             (x=2 & y=3) |
                             (x=3 & y=4)  ).

all x y (neighbor(x,y) <-> successor(x,y) | successor(y,x)).
```

- Remaining clues

```
successor(Gray, Green).
neighbor(Rice, Horse).
neighbor(Beans, Mouse).
```

Programs

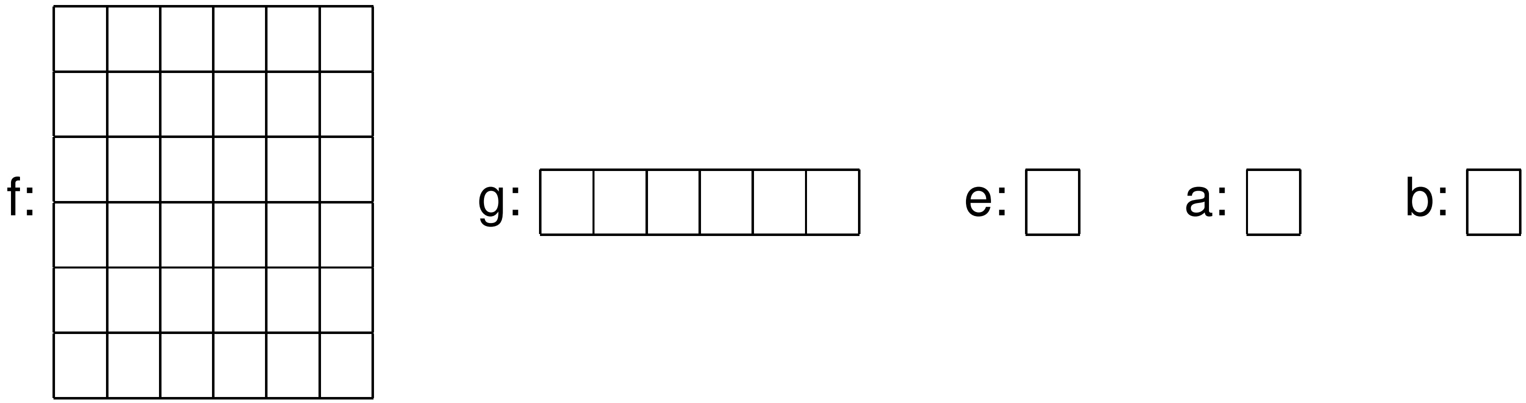
- MACE
  - Reduction to a purely propositional problem.
  - Good performance if formulas are not complicated.
  - Language (and input files) same as for our theorem prover Otter.
  - `www.mcs.anl.gov/AR/mace`
- SEM
  - Reduction to a ground problem with equality.
  - Good performance for equational problems.
  - Can handle sorted logic; that is, several disjoint domains.
  - `www.cs.uiowa.edu/~hzhang/sem.html`

Noncommutative Group – Basic Approach

Generate instances over a domain of $\{$0,1,2,3,4,5$\}$.

f(e,x)=x	f(g(x),x)=e	f(f(x,y),z)=f(x,f(y,z))	f(a,b)$\neq$f(b,a)
f(e,0)=0	f(g(0),0)=e	f(f(0,0),0)=f(0,f(0,0))	f(a,b)$\neq$f(b,a)
f(e,1)=1	f(g(1),1)=e	f(f(1,1),1)=f(1,f(1,1))	
f(e,2)=2	f(g(2),2)=e	f(f(1,0),1)=f(1,f(0,1))	
f(e,3)=3	f(g(3),3)=e	f(f(0,1),1)=f(0,f(1,1))	
f(e,4)=4	f(g(4),4)=e	f(f(0,0),1)=f(0,f(0,1))	
f(e,5)=5	f(g(5),5)=e	f(f(1,1),0)=f(1,f(1,0))	
		f(f(0,1),0)=f(0,f(1,0))	
		f(f(1,0),0)=f(1,f(0,0))	
		f(f(2,2),2)=f(2,f(2,2))	
		. . .	
		(207 more)	

Noncommutative Group – Possible Value Clauses



f(0,0)=0 $\vee$ f(0,0)=1 $\vee$ f(0,0)=2 $\vee$ f(0,0)=3 $\vee$ f(0,0)=4 $\vee$ f(0,0)=5.

$\cdots$ (35 more)

g(0)=0 $\vee$ g(0)=1 $\vee$ g(0)=2 $\vee$ g(0)=3 $\vee$ g(0)=4 $\vee$ g(0)=5.

$\cdots$ (5 more)

e=0 $\vee$ e=1 $\vee$ e=2 $\vee$ e=3 $\vee$ e=4 $\vee$ e=5.

a=0 $\vee$ a=1 $\vee$ a=2 $\vee$ a=3 $\vee$ a=4 $\vee$ a=5.

b=0 $\vee$ b=1 $\vee$ b=2 $\vee$ b=3 $\vee$ b=4 $\vee$ b=5.

Noncommutative Group – Decision Methods

- Given the instances of the input and the possible-value clauses,
  - this is a ground first-order (decidable) problem;
  - is there a practical decision procedure?
- What about splitting (case analysis) on the possible-value clauses with ground completion/resolution?
  - This works,
  - but it doesn't take advantage of *negative constraints*.

Negative Constraints

Negative constraints help, because of the possible-value clauses,

$f(1,2)=0 \ \vee \ f(1,2)=1 \ \vee \ f(1,2)=2 \ \vee \ f(1,2)=3 \ \vee \ f(1,2)=4 \ \vee \ f(1,2)=5.$

- Consider a negative clause that might arise in the noncommutative group example, $f(1,2) \neq f(2,1)$.
  - If we assert $f(2,1)=3$, ordinary deduction will produce $f(1,2) \neq 3$.
  - This is simple enough so that the negative constraint is propagated.
- But consider a more complicated denial, $f(g(4),1) \neq 3$:
  - If we assert $f(2,1) = 3$, we can infer $g(4) \neq 2$, *but our ordinary deduction rules do not do this*.

Clause Transformations

Consider the preceding example (SEM does this):

$$\frac{\begin{array}{c}\text{f(g(4),1)}\neq 3\\ \text{f(2,1)} = 3\end{array}}{\text{g(4)}\neq 2}$$

Transform f(g(4),1)$\neq$3 into g(4)$\neq x$ $\vee$ f(x,1)$\neq$3,
then take the instances over the domain:

g(4)$\neq$0 $\vee$ f(0,1)$\neq$3
g(4)$\neq$1 $\vee$ f(1,1)$\neq$3
$\boxed{\text{g(4)}\neq 2 \ \vee\ \text{f(2,1)}\neq 3}$
g(4)$\neq$3 $\vee$ f(3,1)$\neq$3
g(4)$\neq$4 $\vee$ f(4,1)$\neq$3
g(4)$\neq$5 $\vee$ f(5,1)$\neq$3

Ordinary resolution inference:

$$\frac{\begin{array}{c}\text{g(4)}\neq 2 \ \vee\ \text{f(2,1)}\neq 3\\ \text{f(2,1)} = 3\end{array}}{\text{g(4)}\neq 2}$$

MACE does this kind of transformation *in the extreme*.

MACE Transformations

Consider associativity of f: f(f(x,y),z) = f(x,f(y,z))

- Transform it into a relational form:

  f(x,y)$\neq u$ $\vee$ f(y,z)$\neq v$ $\vee$ f(x,v)$\neq w$ $\vee$ f(u,z)=w.

- Take instances. *This gives us* $6^6 = 46656$ *instances!*

MACE does this kind of transformation for all clauses, then generates and stores all instances as propositional clauses.

MACE Transformations

- Advantages
  - All equality literals are of the form f(2,3)=4 or f(2,3)$\neq$4,
    $\therefore$ we don't need any inference rules for equality,
    $\therefore$ we have a ground resolution problem,
    $\therefore$ we can use a propositional decision procedure (MACE uses DPLL)
  - Negative constraints are propagated as much as possible.
- Disadvantage: *memory requirements!*

Negative Constraints in MACE

- MACE can make this inference propositionally:

$$\frac{\begin{array}{c} f(0,1)\neq 3 \;\vee\; f(1,2)\neq 4 \;\vee\; f(0,4)\neq 5 \;\vee\; f(3,2)=5 \\ f(0,1) = 3 \\ f(0,4) = 5 \\ f(3,2)\neq 5 \end{array}}{f(1,2)\neq 4}$$

- Without the transformation, we have to make a backward equational inference and a negative inference:

$$\frac{\begin{array}{c} f(f(0,1),2) = f(0,f(1,2)) \\ f(0,1) = 3 \end{array}}{f(3,2) = f(0,f(1,2))} \qquad \frac{\begin{array}{c} f(3,2) = f(0,f(1,2)) \\ f(3,2)\neq 5 \end{array}}{f(0,f(1,2))\neq 5} \qquad \frac{\begin{array}{c} f(0,f(1,2))\neq 5 \\ f(0,4) = 5 \end{array}}{f(1,2)\neq 4}$$

Hybrid Methods

We are developing a hybrid method that

- works like MACE (complete transformation) for the simple clauses,
- does a limited amount of transformation on complex clauses,
- does ground equational reasoning,
- propagates negative constraints with inference rules.

We still have (recursive) case analysis on “possible-value clauses”.

$\therefore$ backtracking,

$\therefore$ incremental, undoable operations.

Isomorphism Reduction

- Applies to both MACE and SEM methods.
- Consider the search for noncommutative groups of order 6.
  - We have 3 constants: {e,a,b}.
  - Without loss of generality, they can be distinct, so we can assign them to members of the domain, say {0,1,2}.
  - This reduces the number of models found by MACE from 2160 to 18.
- If we cannot assume uniqueness, we can add constraints:
  $e = 0$
  $a = 0 \lor a = 1$
  $a = 0 \rightarrow (b = 0 \lor b = 1)$
  $a = 1 \rightarrow (b = 0 \lor b = 1 \lor b = 2)$
  but this list grows exponentially in the number of constants.

Isomorphism Reduction - LNH

- Least Number Heuristic [SEM].
  - Say we are focusing on cell f(2,3). We can partition the domain into two sets: *constrained* and *unconstrained*.

    A member is constrained if it is mentioned in any of the current assignments. For example, f(1,2)=4 constrains 1, 2, and 4.
  - We must try all constrained values and the smallest of the unconstrained values.
- This constrains all values early in the search, but it helps.
- We probably cannot eliminate all isomorphism.
  - challenge: method that finds exactly one noncomm. group of order 6.

Conclusions

- Finite model search works!
  - answering open questions in math and logic
  - debugging axiom systems
- There should be important applications in other areas.
  - design problems
  - debugging design specifications
- The future looks good for more powerful methods.