

Theorem Proving in the Computer Science World

Christelle Scharff, PhD

Pace University, New York
Maths-CS seminar, 04/02/2003



Outline

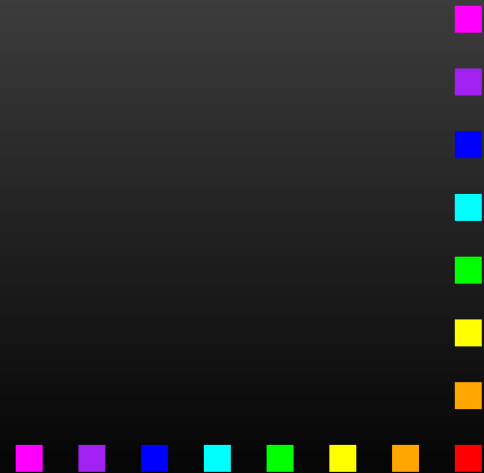
- What is theorem proving?
- What are the proving techniques?
- What is a theorem prover?
- Theorem proving in the real world
- Rewriting proofs or valley proofs
 - Some definitions and notations
 - The Word problem
 - Rewriting and all that
 - Completion
- My research

What is theorem proving?



What are the proving techniques?

- Induction proofs
- Contradiction proofs
- Saturation proofs / Compilation proofs
- Rewriting proofs / Valley proofs



What is theorem proving?

- Theorem proving = Automated deduction = Automated reasoning
- It is concerned with the mechanization of the deductive process in its fullest meaning [Loveland, 99].
- Active area of research since the 1950s. The more recent achievements are software.

Mechanization of reasoning

- Aristotle Syllogism
All men are mortal.
Socrates is a a man.
Therefore, Socrates is mortal.
- Formalization of the problem - Which logic to reason in?
Hypotheses: $Man(x) \Rightarrow Mortal(x)$ and $Man(Socrates)$
- Goal to prove: *$Mortal(Socrates)$*
- Prove the goal - How?????????

What is a theorem prover or deductive software?



What is a theorem prover?

Theorem prover = Inference rules + Proof strategies

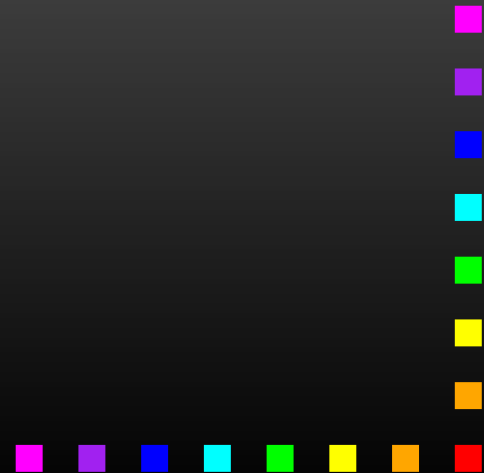
- **Inference rules:** How to deduce new data? How to remove data?
- **Strategies:** How to apply inference rules? Interactively? Automatically? Expansion strategies? Simplification strategies?
[Bonacina, 99]



Properties

- **Soundness**
Do not prove that True = False
Proving only true formulas.
- **Completeness**
If a formula is true, it can be proved.

Syntax/Semantics



Challenges

- The basic infrastructure of theorem provers requires investment in:
 - syntactic tools (parsers, typecheckers,...),
 - primitive operations (substitution, matching, unification...),
 - advanced operations (constraint satisfaction, rewriting, decision procedures...)
- Many of the problems are undecidable or if decidable, algorithms are exponential.

Theorem proving in the real world

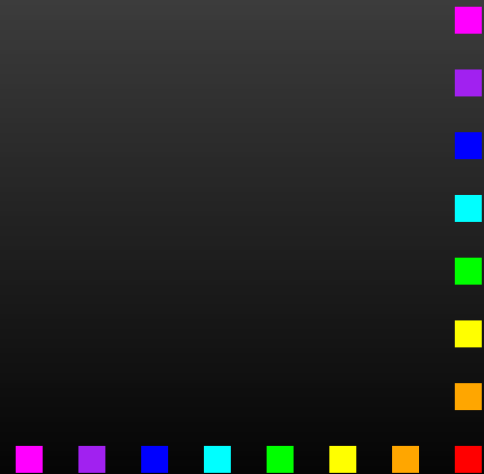
- Assist mathematicians
 - Moufang identities in alternative rings - **SBREVE** [Anantharaman, Hsiang,90]
 - Robbins algebras - Robbins algebra are boolean **EQP** [McCune,97]
- Software verification
 - Critical applications - **PVS**
 - Authentication protocols - **Isabelle, ELAN, DATAC**



- Hardware verification
 - Pentium chip - RRL [Kapur, Subramaniam,98]
- Artificial intelligence (knowledge representation, expert systems)
- Programming language paradigm (logic programming, functional programming, rewrite programming)



Rewriting proofs/Valley proofs



What logic to reason in?



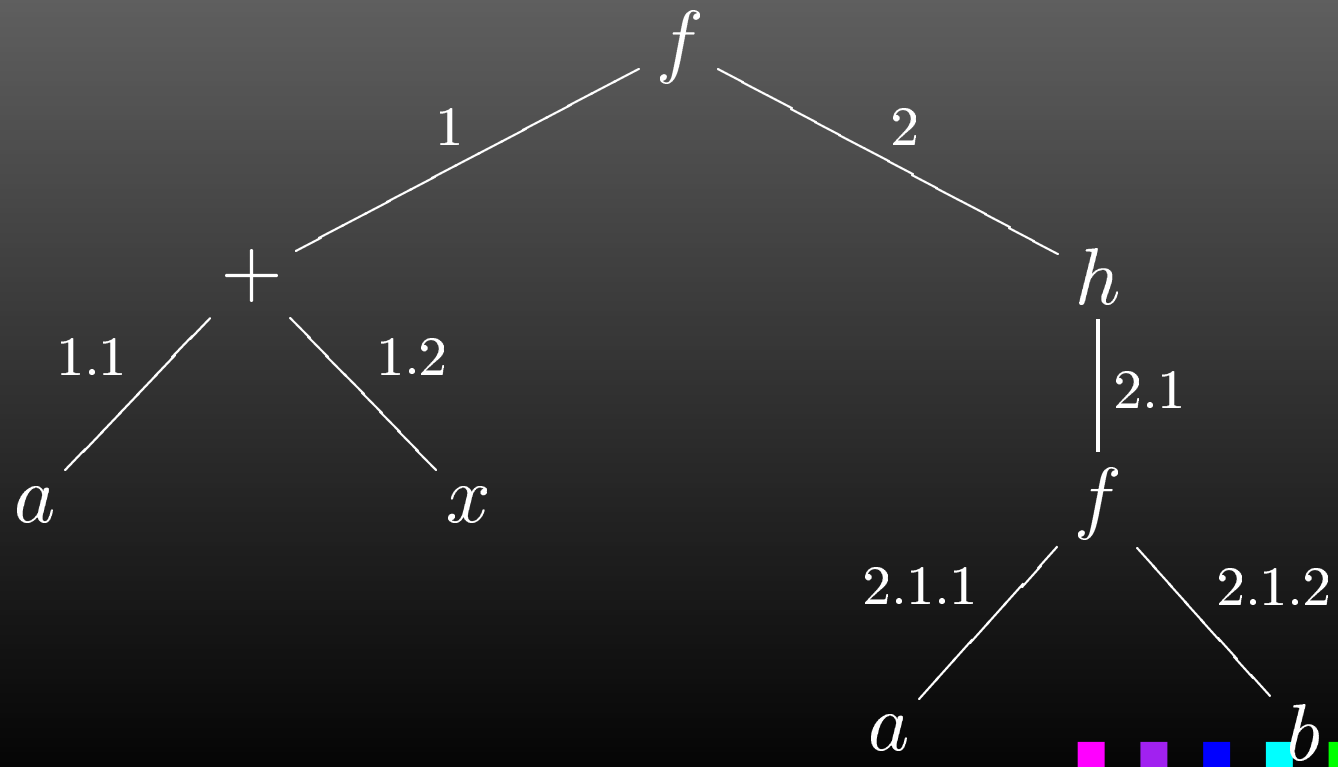
Terms

- A term is
 - a variable ($x, y, z\dots$)
 - a constant ($a, b, c\dots$)
 - $f(t_1, \dots, t_n)$ where f is a n -ary function symbol and each t_i is a term.
- A ground term ($f(a), g(c, d)\dots$) has no variables.
- $u[s]$: The term u contains the term s .



Example

- $f(a + x, h(f(a, b)))$ is represented by the tree:



Equalities

- \approx is a congruence relation i.e.
 1. *Reflexive*: $x \approx x$
 2. *Symmetric*: $x \approx y \rightarrow y \approx x$
 3. *Transitive*: $x \approx y$ and $y \approx z \rightarrow x \approx z$
 4. *Congruence*:
 $x_1 \approx y_1$ and ... and $x_n \approx y_n \rightarrow f(x_1, \dots, x_n) \approx f(y_1, \dots, y_n)$
- **Examples**: $\forall x, y, (x + y \approx y + x)$ (C),
 $\forall x, y, z, ((x + y) + z) \approx x + (y + z)$ (A),
 $f(a) \approx b, \forall x, (f(x) \approx a)$

First order logic

- **First order logic**

$\forall x, y, (\text{parent}(x, y) \text{ and } \text{parent}(y, z) \Rightarrow \text{grandparent}(x, z))$

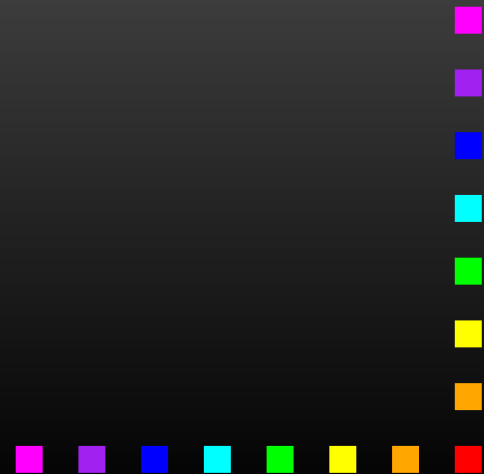
- **First order logic with equality**

$\forall x, (x \neq 0 \Rightarrow \exists y, (y \approx 1/x))$

- **First order logic with equalities ONLY.**



How to prove a theorem?

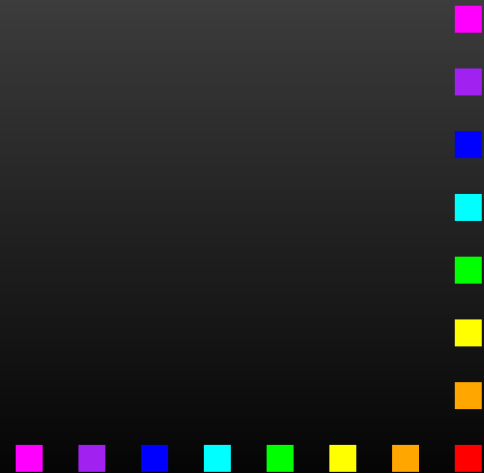


The Word problem

- Given a set of equalities E and a goal $s \approx t$, is $s \approx t$ true in all models of E ?

Word problem is undecidable.

- **Goal: Find a method which OFTEN gives a decision procedure.**



Solutions to the Word problem

- **Solution 1:** Generate all equalities implied by E using inference rules and strategies.
Disadvantage of solution 1: Generation of all equalities may not halt, not goal oriented.
- **Solution 2:** Apply equalities to goal until both sides are the same.
Benefit of solution 2: Goal oriented.
Disadvantage of solution 2: May not halt if goal is false.



Example

- Group axioms:

$$x + e \approx x$$

$$x + i(x) \approx e$$

$$(x + y) + z \approx x + (y + z)$$

- How to prove that: $x + e \approx e + x$?

- $$\begin{aligned} e + x &\approx e + (x + e) \approx e + (x + (i(x) + i(i(x)))) \\ &\approx e + (x + i(x)) + i(i(x)) \approx e + (e + i(i(x))) \\ &\approx (e + e) + i(i(x)) \approx e + i(i(x)) \approx (x + \\ & i(x)) + i(i(x)) \\ &\approx x + (i(x) + i(i(x))) \approx x + e \end{aligned}$$

Rewriting

- Using equalities with an orientation using an ordering
If $s \approx t$ and $s \succ t$, then we write $s \rightarrow t$.
- An ordering \succ is a binary relation:
 - **Reflexive:** $x \succ x$
 - **Transitive:** $x \succ y$ and $y \succ z \Rightarrow x \succ z$
- C cannot be oriented. A can be oriented into $(x + y) + x \rightarrow x + (y + z)$.



A simple rewrite system

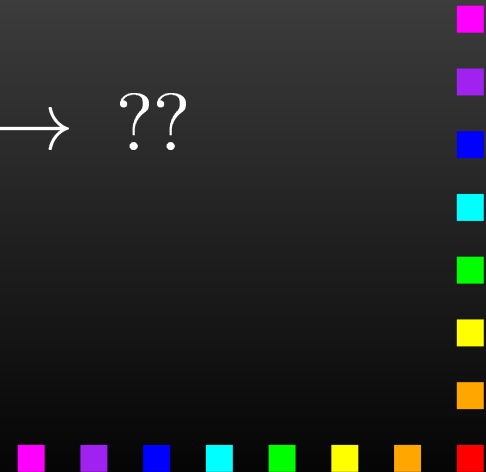
$$\bullet \bullet \rightarrow \circ$$

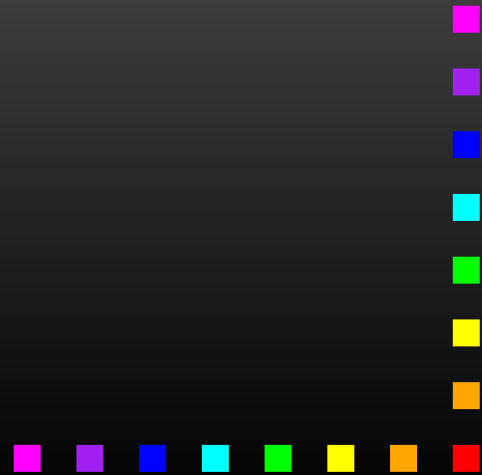
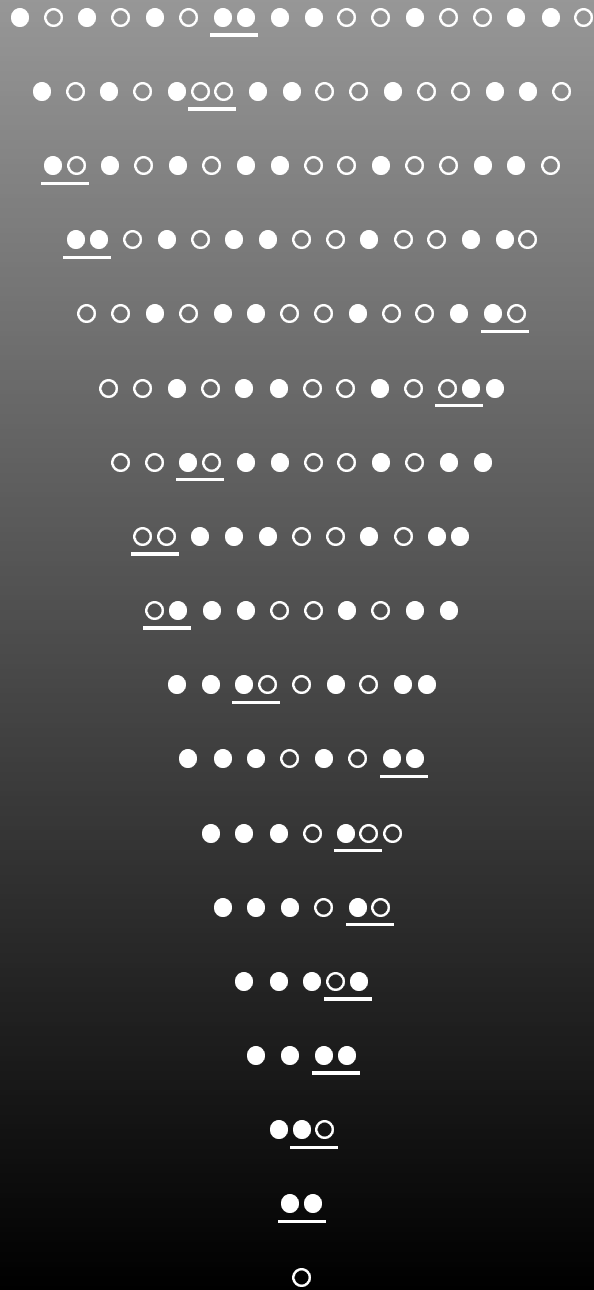
$$\circ \circ \rightarrow \circ$$

$$\bullet \circ \rightarrow \bullet$$

$$\circ \bullet \rightarrow \bullet$$

$$\bullet \circ \bullet \circ \bullet \circ \bullet \bullet \bullet \bullet \circ \circ \bullet \circ \circ \bullet \bullet \circ \rightarrow ??$$





Differentiation rewrite system

- The differentiation operator D with respect to a variable X

- $D(X) \rightarrow 1$

$$D(C) \rightarrow 0$$

$$D(x + y) \rightarrow D(x) + D(y)$$

$$D(x - y) \rightarrow D(x) - D(y)$$

$$D(-x) \rightarrow -D(x)$$

$$D(x * y) \rightarrow (x * D(y)) + (y * D(x))$$

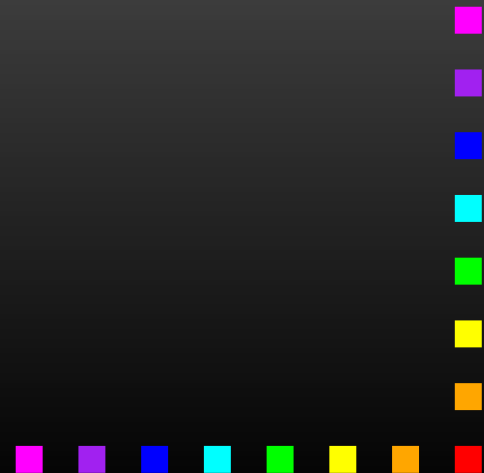
$$D(x/y) \rightarrow D(x)/y - ((x * D(y))/(y * y))$$

$$D(\ln(x)) \rightarrow D(x)/x$$



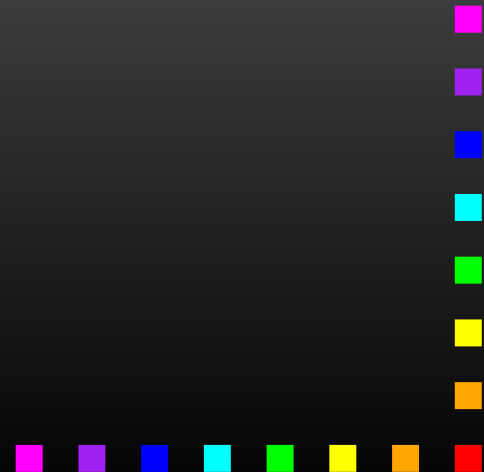
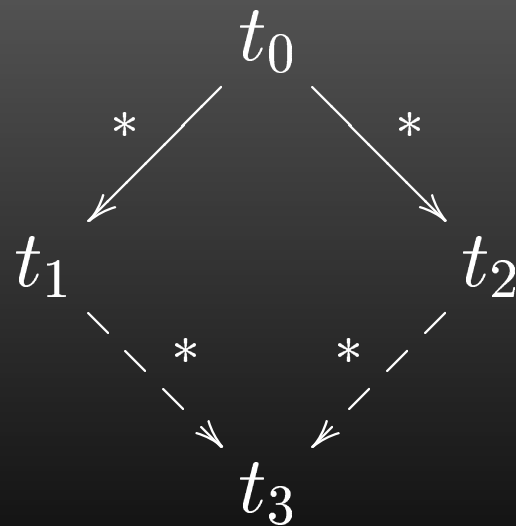
A program

- Append on lists
- $append(nil, y) \rightarrow y$
 $append(cons(x, y), z) \rightarrow cons(x, append(y, z))$



Confluence of rewrite systems

- **Confluence:** If $s \rightarrow^* t$ and $s \rightarrow^* u$, then there exists v such that $t \rightarrow^* v$ and $u \rightarrow^* v$.

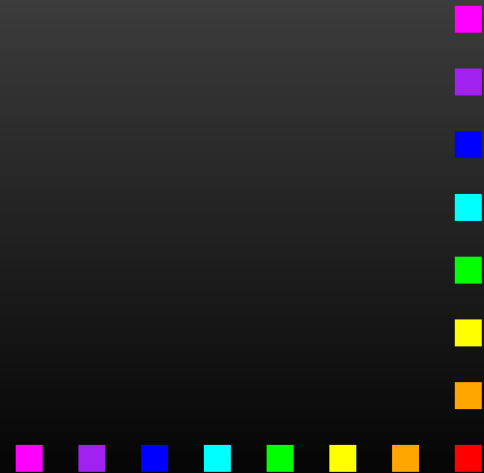


Termination of rewrite systems

- **Termination:** There exists no infinite sequence

$$s_1 \rightarrow \dots \rightarrow s_n \rightarrow \dots$$

Undecidable property.



Completion

- Solves the word problem
- Based on Rewriting.
- Main inference rule: **Critical Pair inference rule**
- **Compilation of E using an ordering and CP**
- Obtaining of a deterministic rewrite system
- Proving is done using a **Rewriting proof (valley proof)**.

$s \approx t$ is true if there exist a term u such that
 $s \rightarrow^* u$ and $t \rightarrow^* u$.



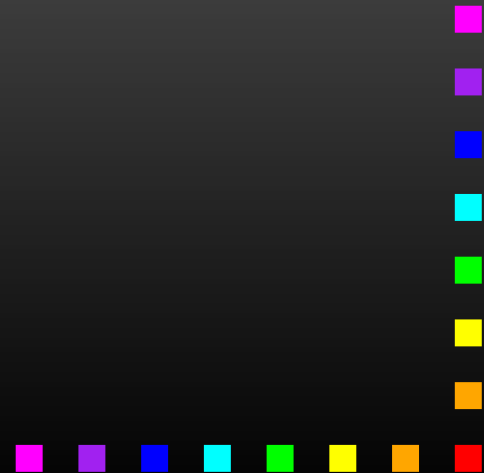
Critical Pair

- $$\frac{g(a) \rightarrow b \quad a \rightarrow c}{g(c) \rightarrow b}$$

$((b, g(c))$ is a critical pair)

- $$\frac{g(x) \rightarrow f(x) \quad g(a) \rightarrow b}{f(a) \rightarrow b}$$

$((f(a), b))$ is a critical pair)



Example

- Group axioms:

$$x + e \approx x$$

$$x + (y + z) \approx (x + y) + z$$

$$x + i(x) \approx e$$

- How to check that:

$$x + e \approx e + x$$



Example (cont)

E is transformed to the rewrite system

$$x + e \rightarrow x$$

$$e + x \rightarrow x$$

$$x + (y + z) \rightarrow (x + y) + z$$

$$x + i(x) \rightarrow e$$

$$i(x) + x \rightarrow e$$

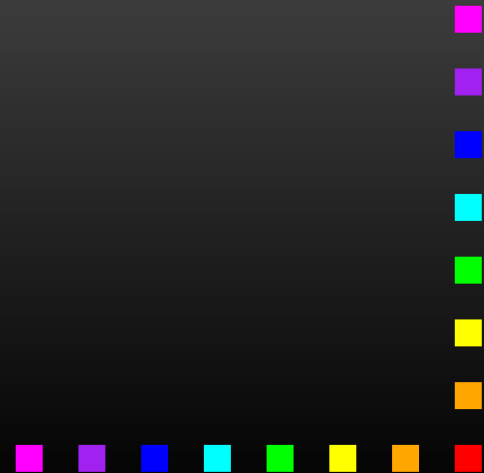
$$i(e) \rightarrow e$$

$$(y + i(x)) + x \rightarrow y$$

$$(y + x) + i(x) \rightarrow y$$

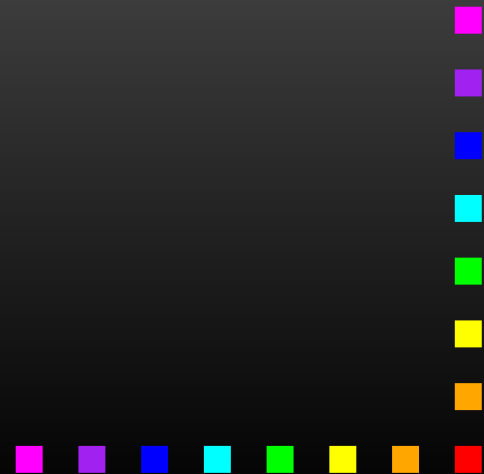
$$i(i(x)) \rightarrow x$$

$$i(x + y) \rightarrow i(y) + i(x)$$



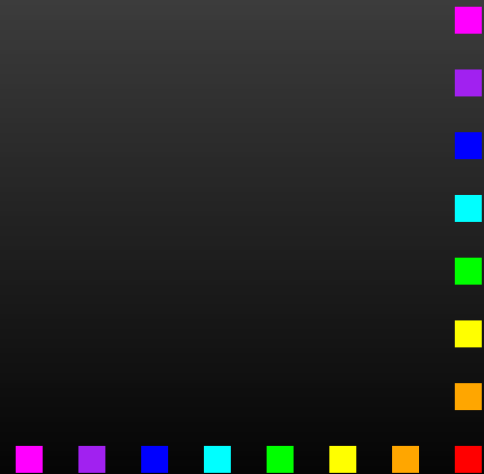
Example (cont)

- The proof of $e + x \approx x + e$ is now obvious.
- The proof of $i(x + y) \approx i(y) + i(x)$ is now obvious.



My research

- Completion modulo with constraints and simplification
- Decision procedures
- 'Little Engines of Proofs' proposal



References

- Logician in the land of OS: Abstract State Machine in Microsoft, Yuri Gurevich, LICS 2001, <http://research.microsoft.com/gurevich>.
- Automated Deduction, Looking ahead, Donald W. Loveland, American association for artificial intelligence, 1999.
- With Major Math Proof Brute Computers Show Flash of Reasoning Power, Gina Kolata, New York Times, December 10th 1996. (The Robbins Problem, McCune).