Theorem Provers using Cooperating Decision Procedures

- Introduced by Nelson and Oppen [TOPLAS 1979]
- Combines decision procedures for a set of disjoint theories, producing a procedure for their union
- Key ideas
  - introduce auxiliary variables to remove mixed application of function symbols
  - theories propagate discovered equalities to each other

Example

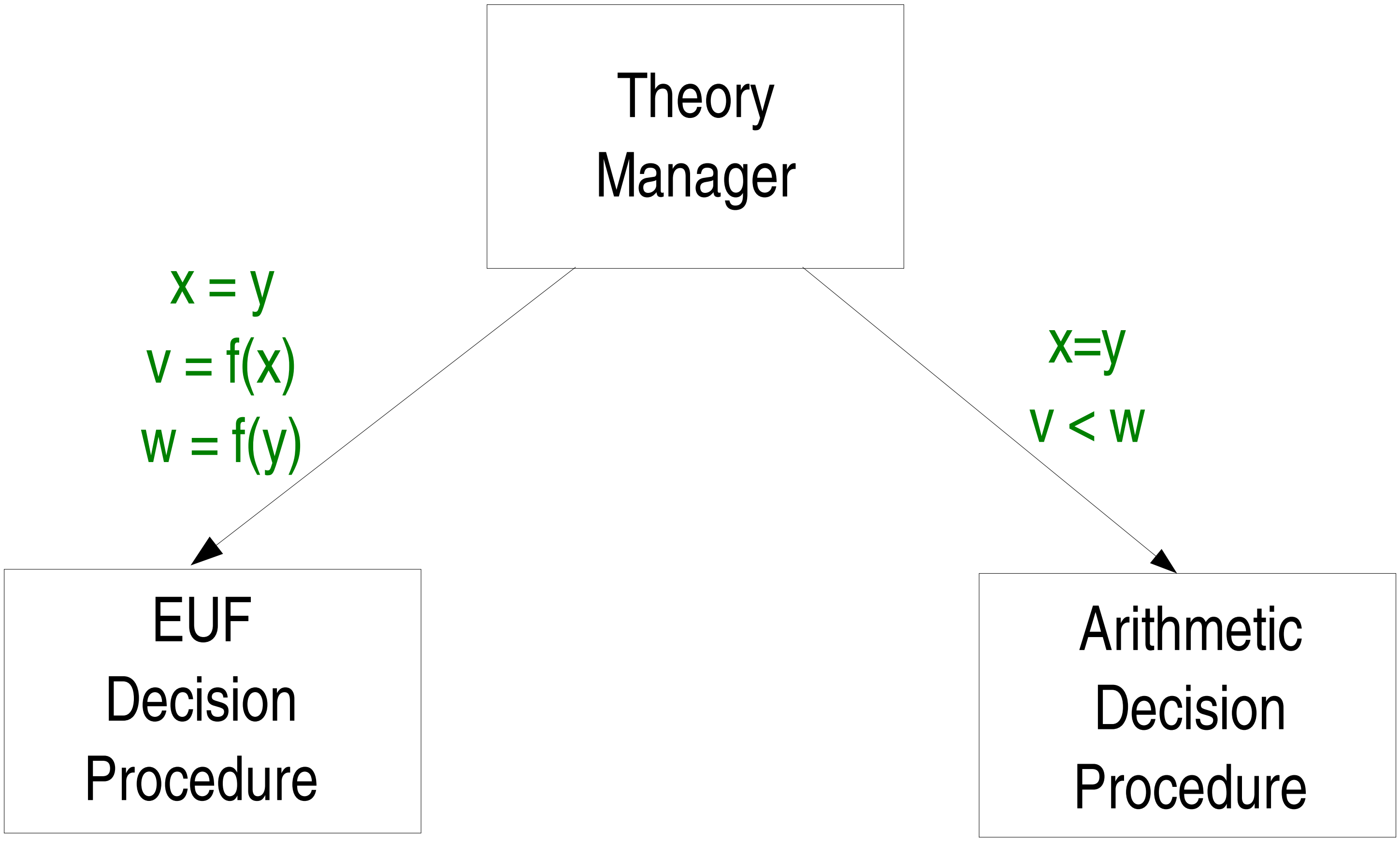
- Suppose we want to check satisfiability of

$$(x = y) \quad \wedge \quad (\ f(x) < f(y)\ )$$

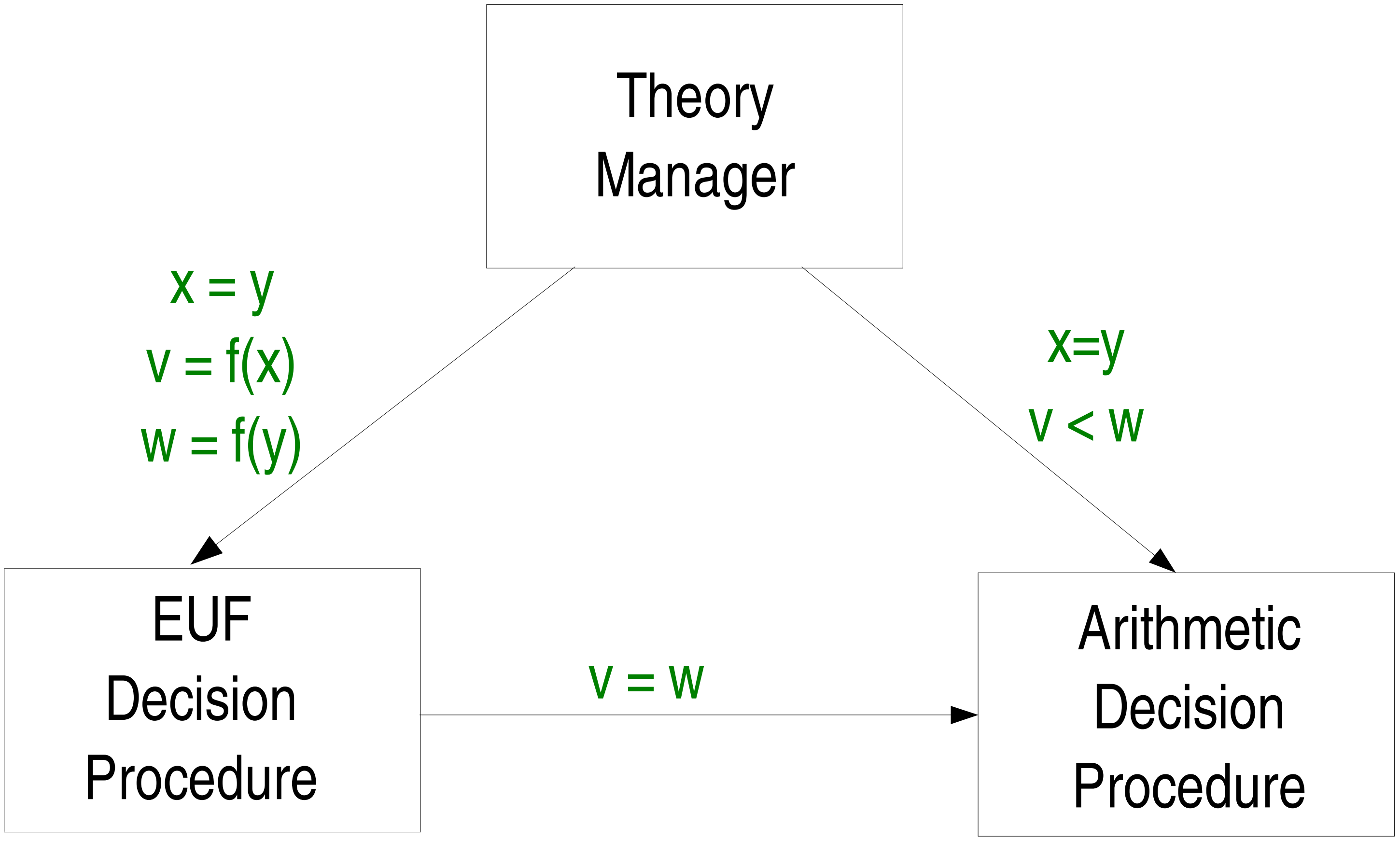
- Introduce auxiliary variables v, w

$$(x = y) \quad \wedge \quad (\ v \ < \ w\ )$$
$$\wedge \quad (v = f(x)) \quad \wedge \quad (w = f(y))$$

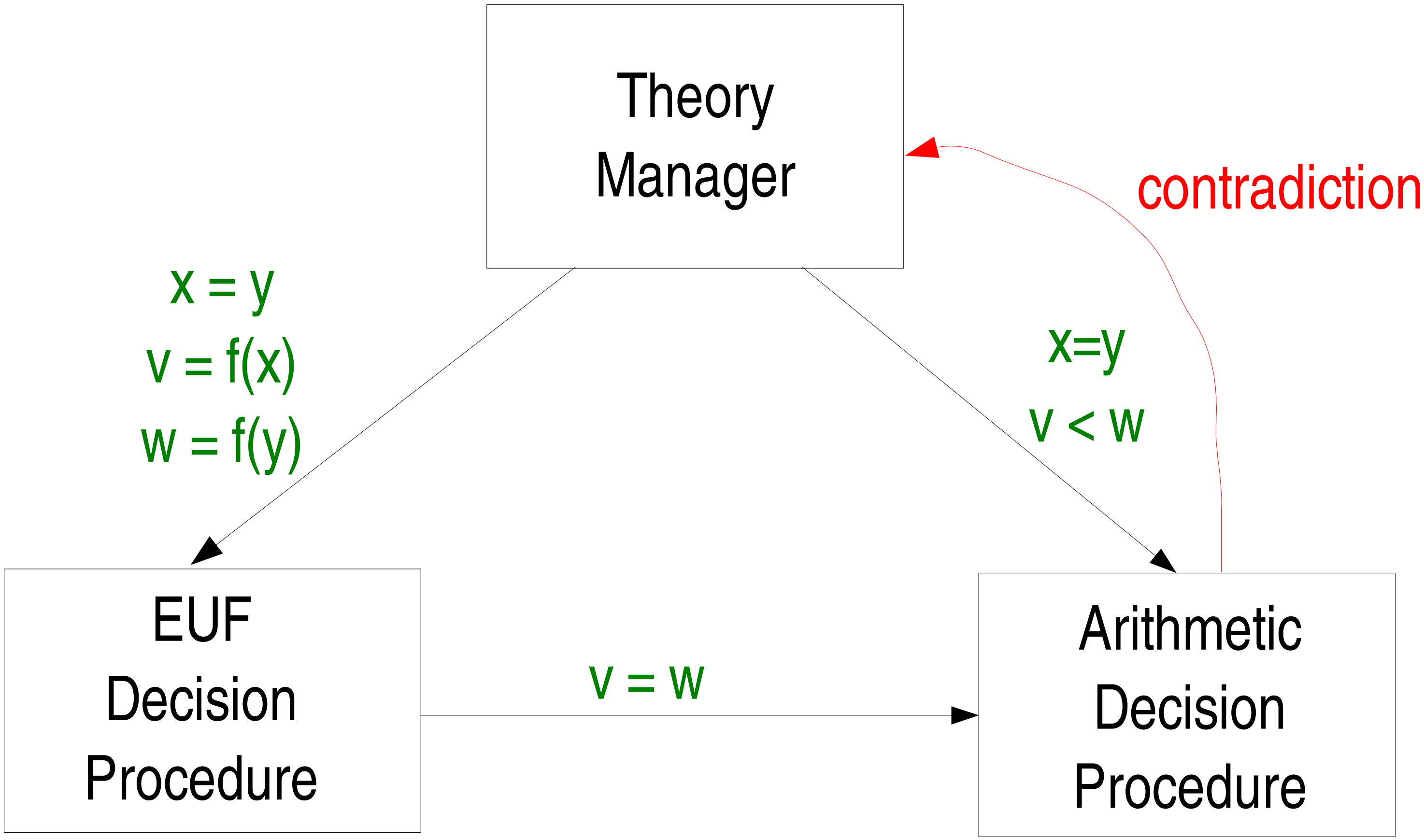
Checking $(x = y) \wedge (f(x) < f(y))$



Checking $(x = y) \wedge (f(x) < f(y))$

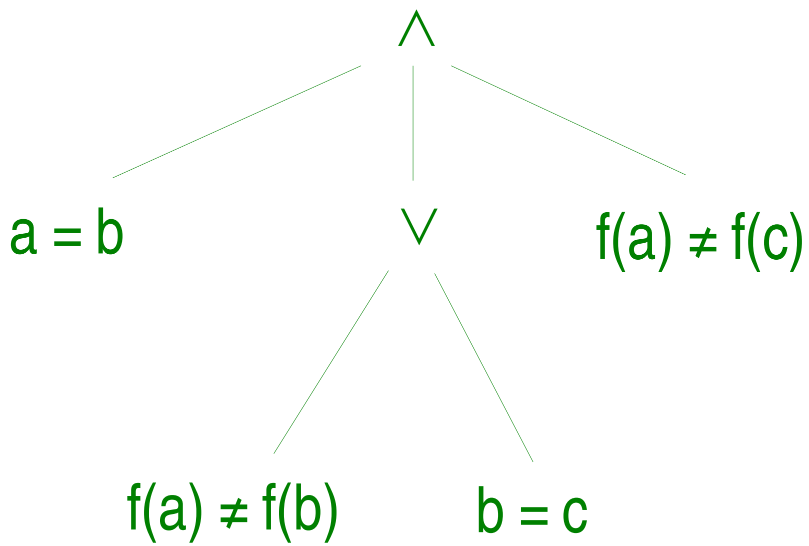


Checking $(x = y) \wedge (f(x) < f(y))$



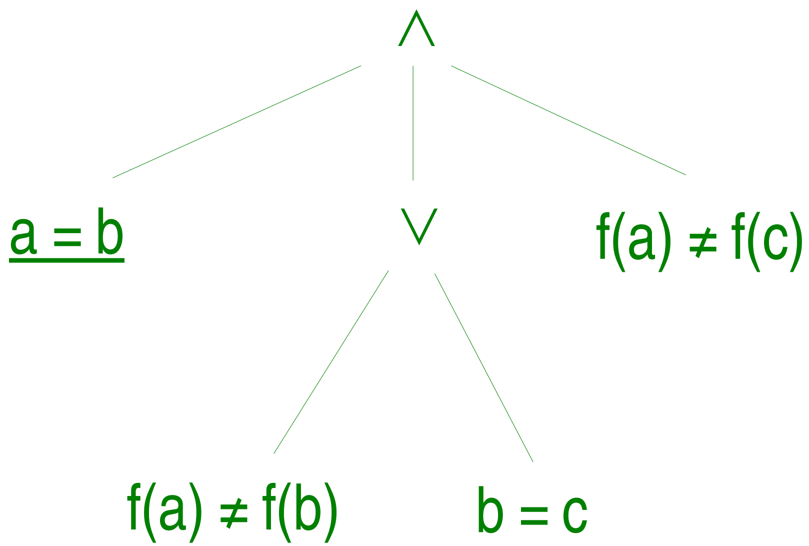
Backtracking in Nelson-Oppen

- Consider



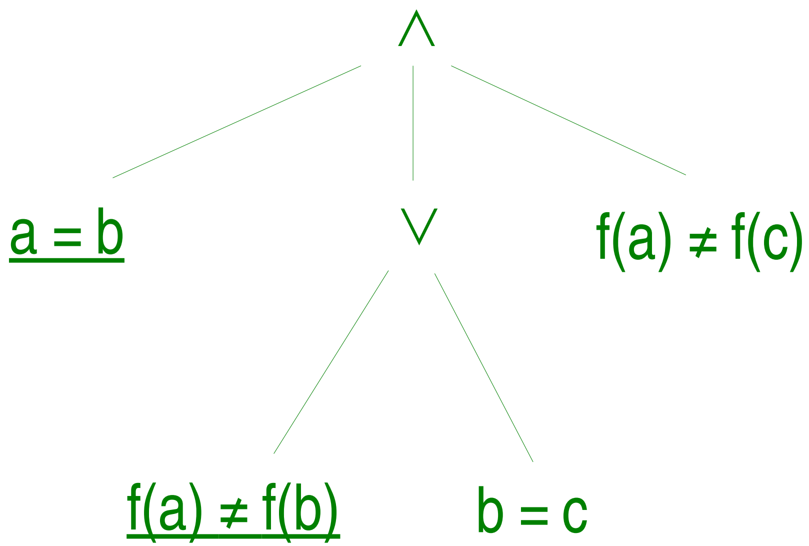
Backtracking in Nelson-Oppen

- Consider



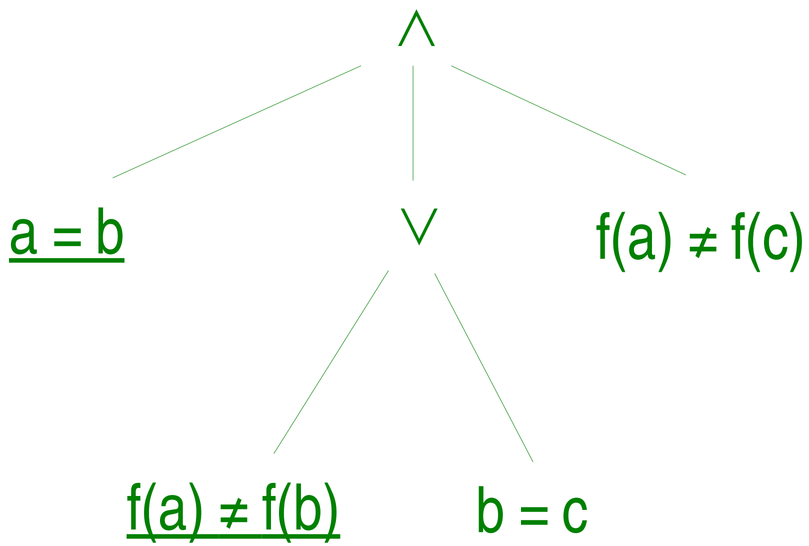
Backtracking in Nelson-Oppen

- Consider



Backtracking in Nelson-Oppen

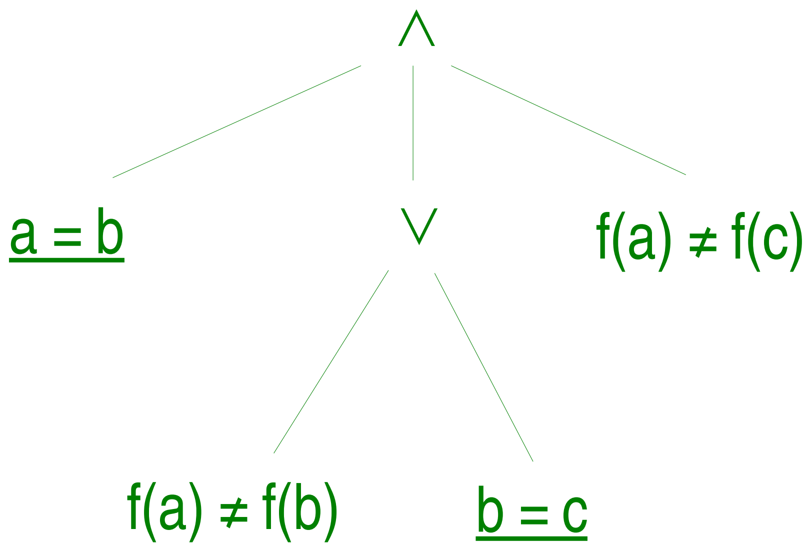
- Consider



Inconsistency detected by the EUF procedure.
So backtrack, and try other branch.

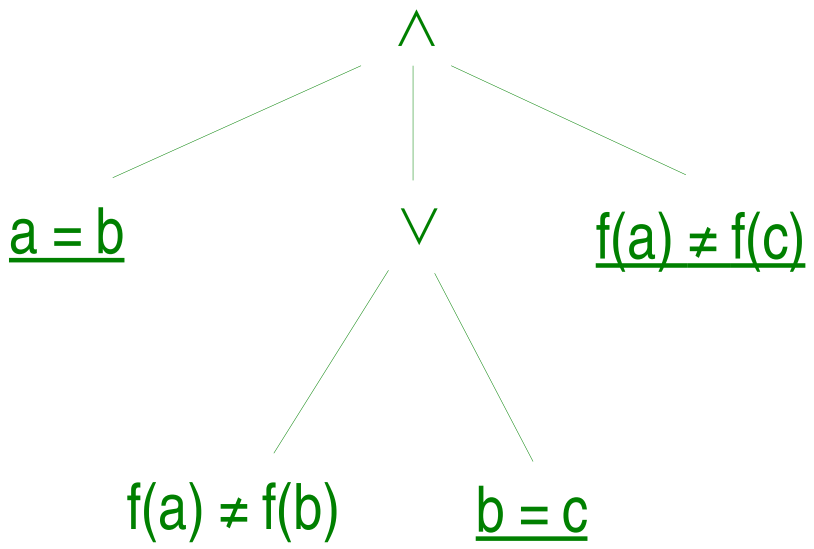
Backtracking in Nelson-Oppen

- Consider



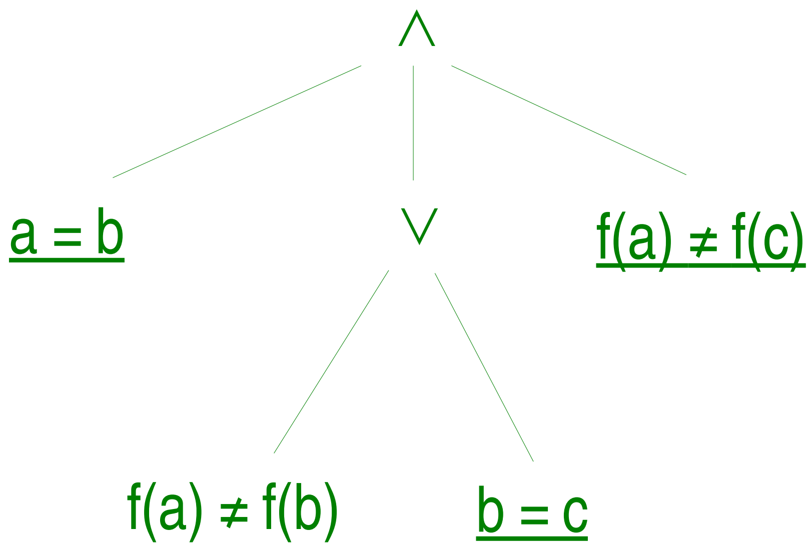
Backtracking in Nelson-Oppen

- Consider



Backtracking in Nelson-Oppen

- Consider



This assignment is also inconsistent with EUF.

There are no branches left, so the formula is unsatisfiable.

Simplify

- Written by Greg Nelson, Dave Detlefs and Jim Saxe
- Supports
  - EUF (using the E-graph data structure)
  - rational linear arithmetic (using the Simplex algorithm)
  - quantified formulae involving $\exists$ and $\forall$ (using matching)
- Very successful: used as the engine in many checkers
  - ESC/Modula-3, ESC/Java, SLAM, ...

Experience with Simplify

- Backtracking search is too slow
  - Far surpassed by recent advances in SAT solving
- Inconsistencies reveal only one bit of information
  - Theory modules repeatedly rediscover the “same” inconsistencies

A Prover using Lazy Proof Explication

- Key ideas
  - use a fast SAT solver to find candidate truth assignments to atomic formulae
  - have theory modules produce compact “proofs” that are added to the SAT problem to reject all truth assignments containing the “same” inconsistency
- Requires
  - proof-explicating theory modules

Example using lazy proof explication

- Suppose we want to check satisfiability of

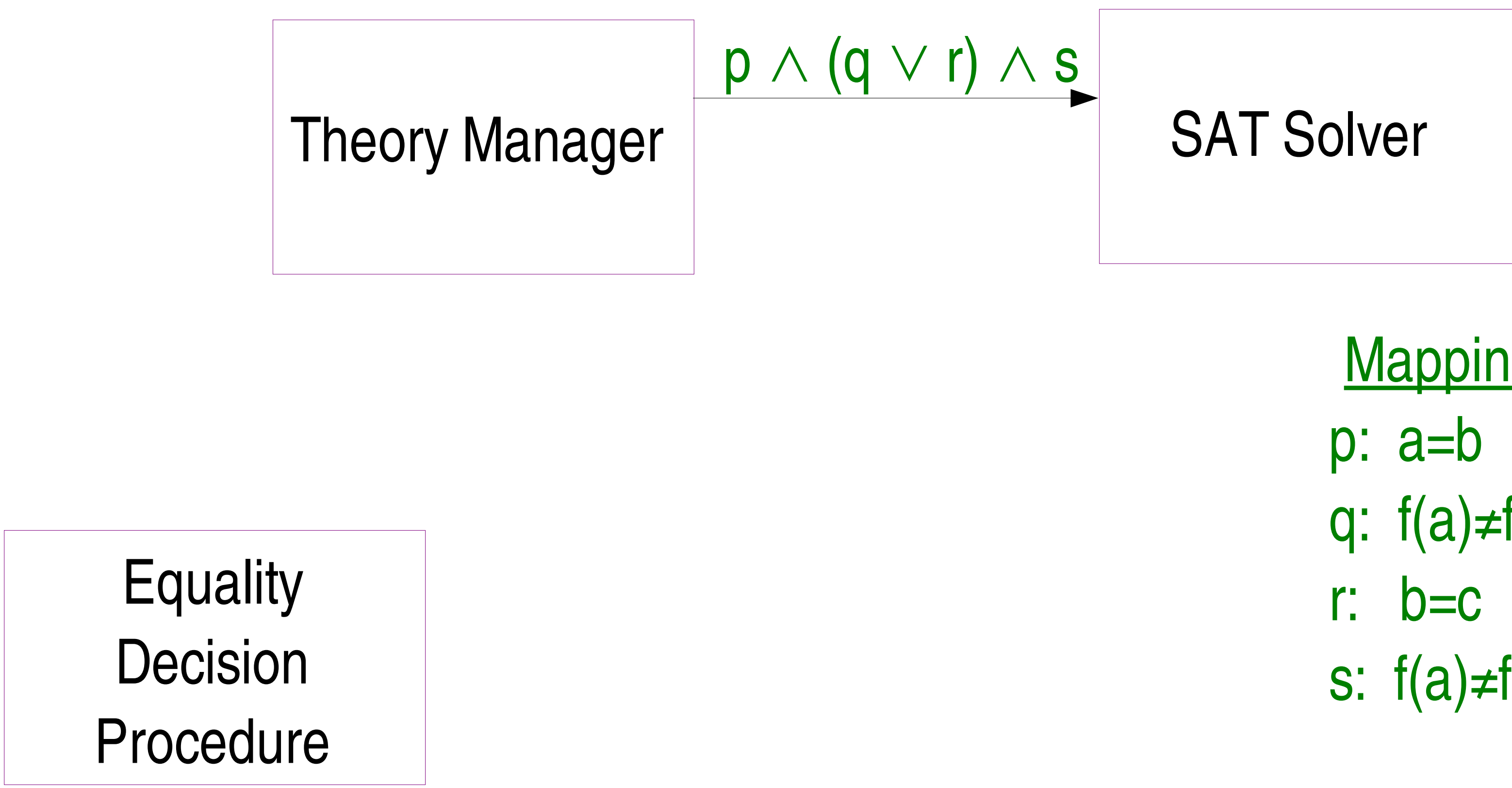
$$(a = b) \quad \wedge \quad (f(a) \neq f(b) \vee b = c) \quad \wedge \quad (f(a) \neq f(c))$$

- Encode it in propositional logic

$$p \quad \wedge \quad (\quad q \quad \vee \quad r \quad) \quad \wedge \quad s$$

where p denotes (a=b), and so on

Example using lazy proof explication



<u>Mapping</u>

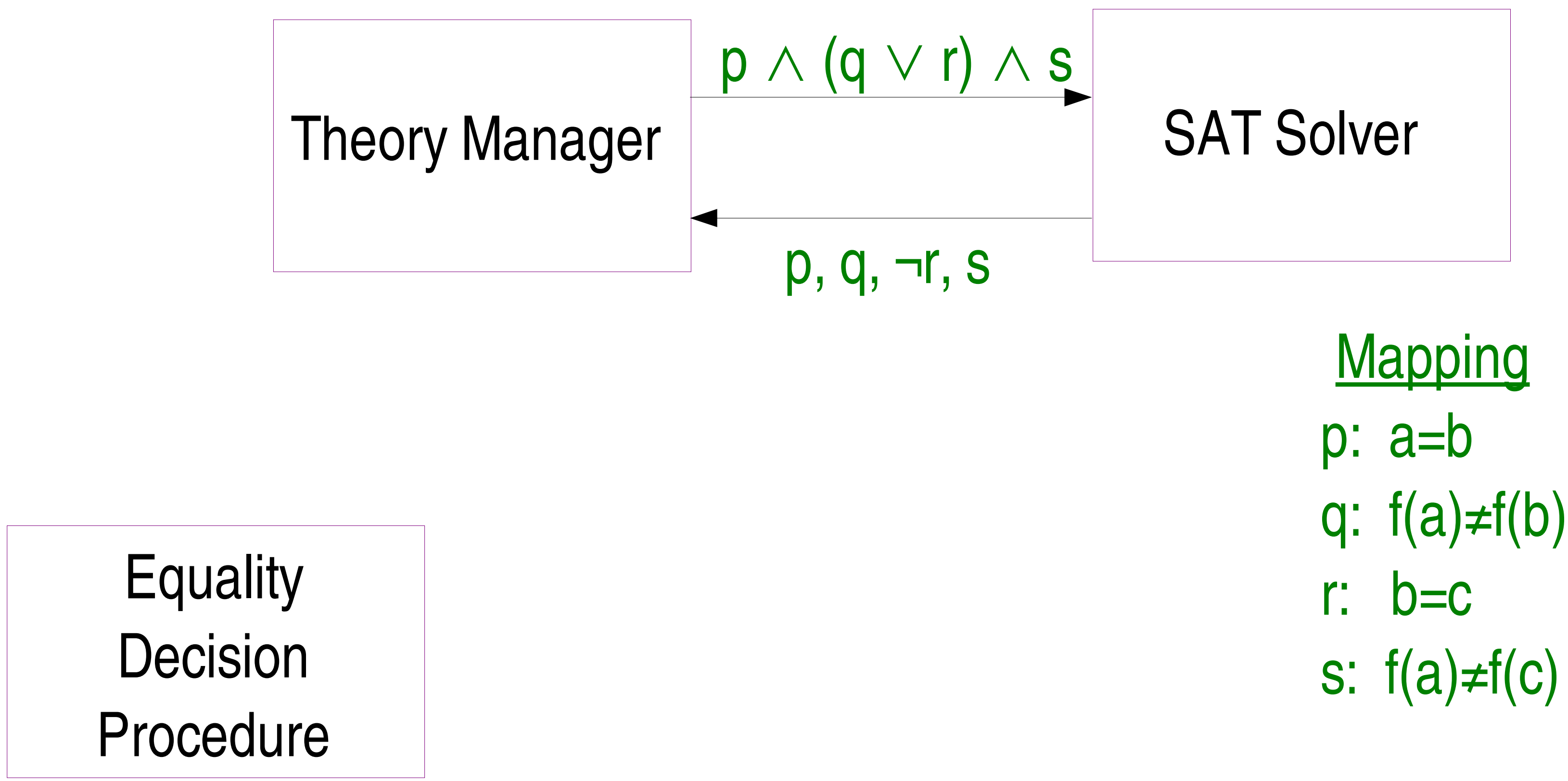
p: a=b

q: f(a)≠f(b)

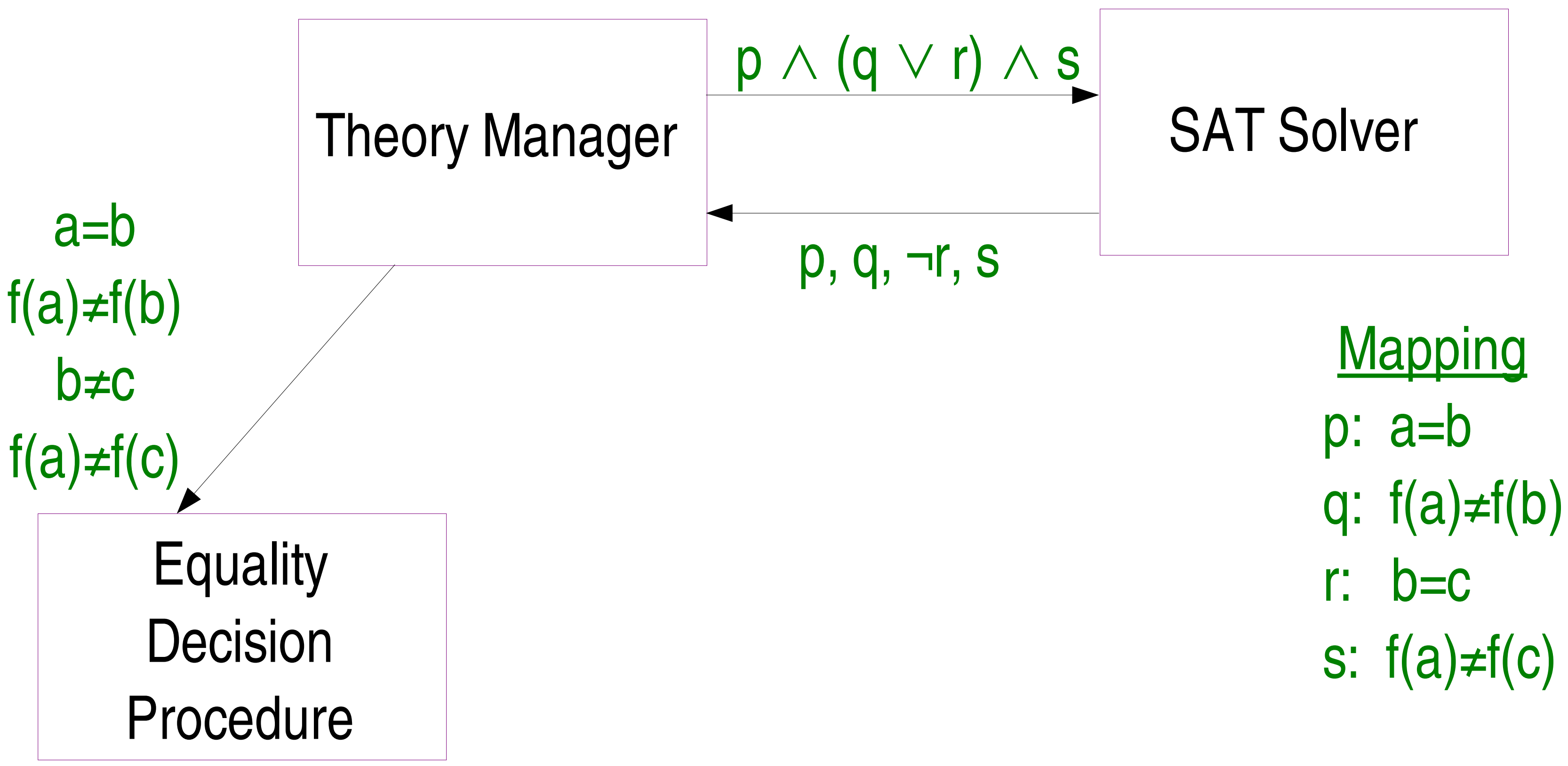
r: b=c

s: f(a)≠f(c)

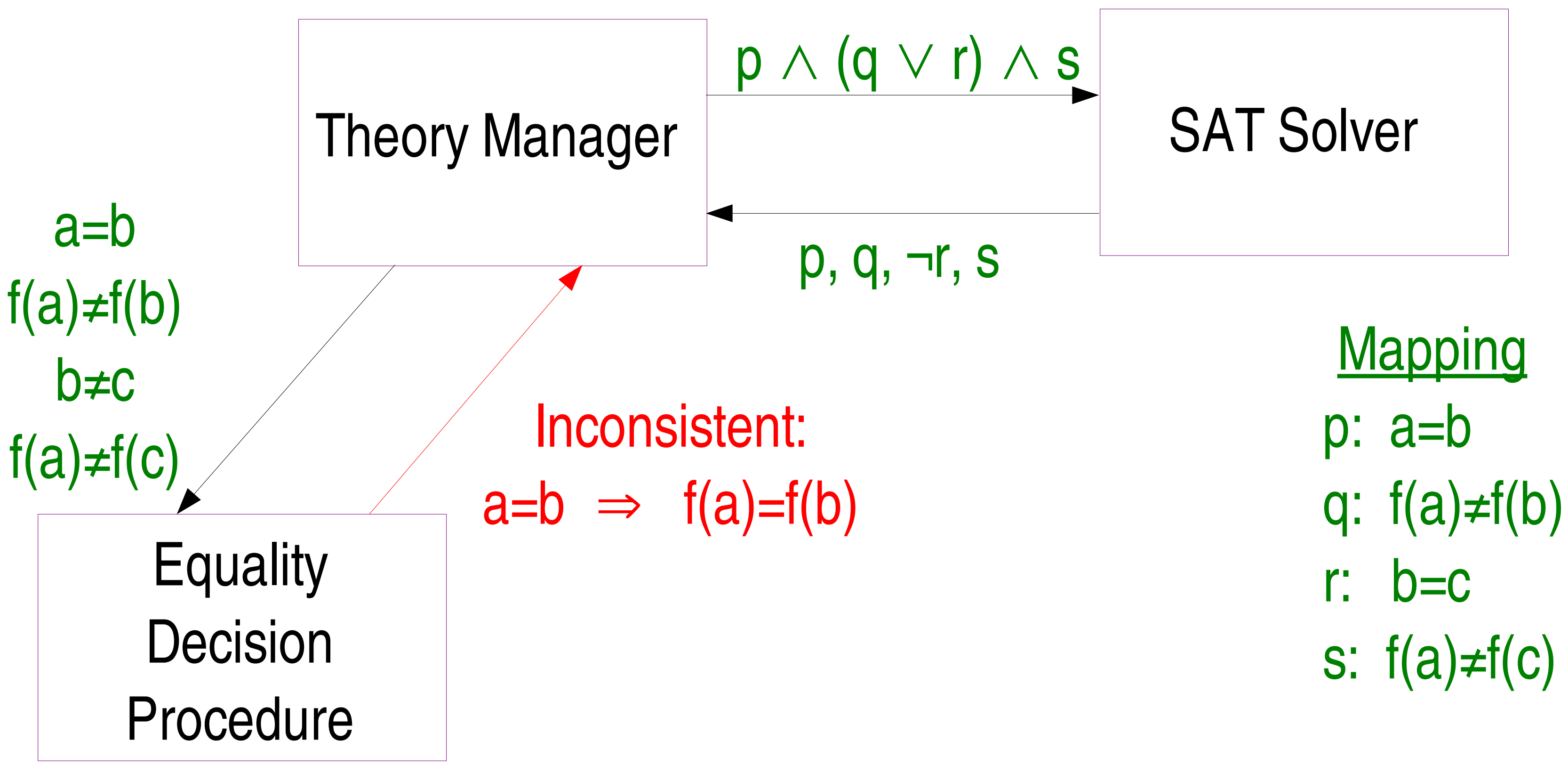
Example using lazy proof explication



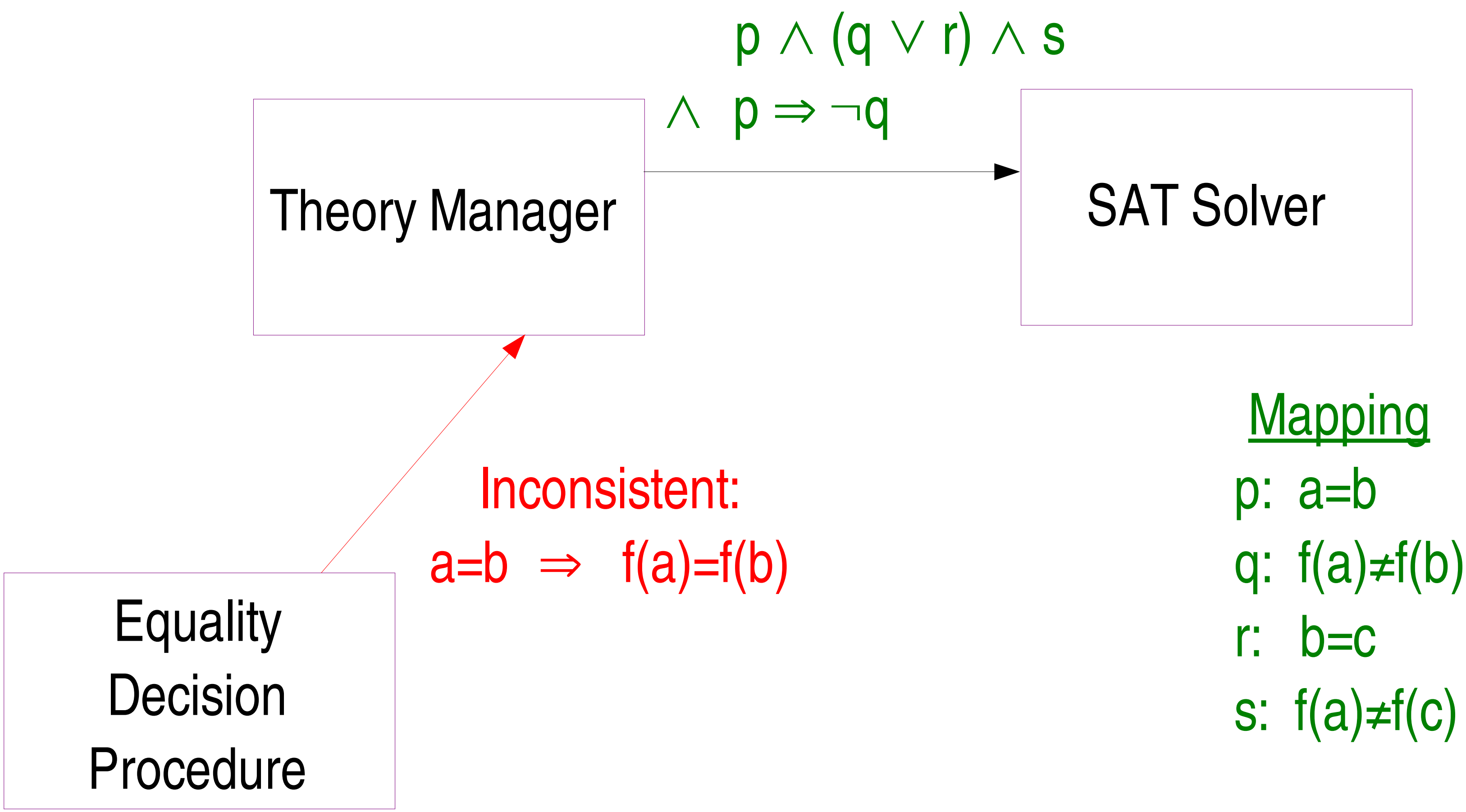
Example using lazy proof explication



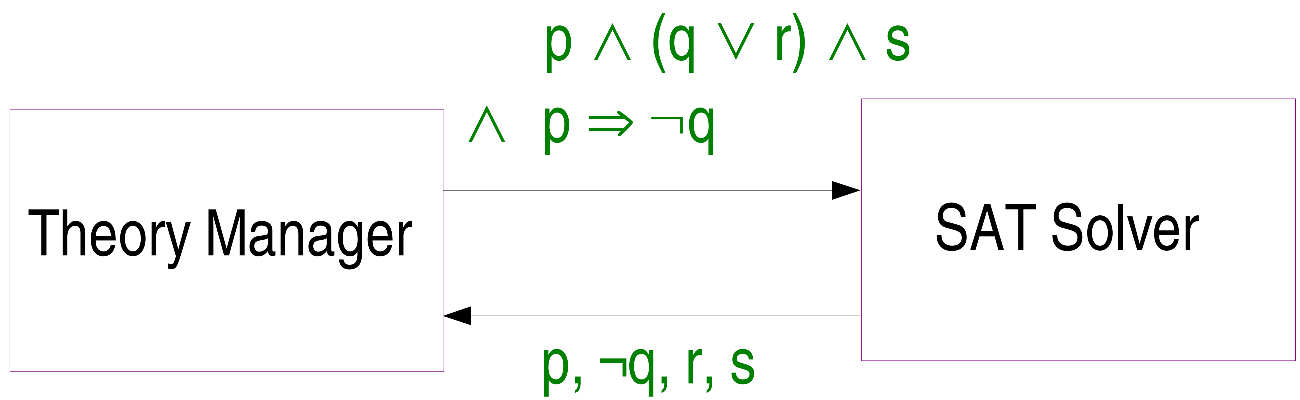
Example using lazy proof explication

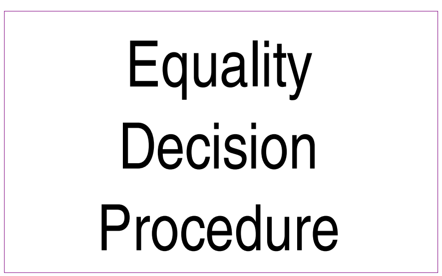


Example using lazy proof explication



Example using lazy proof explication





<u>Mapping</u>

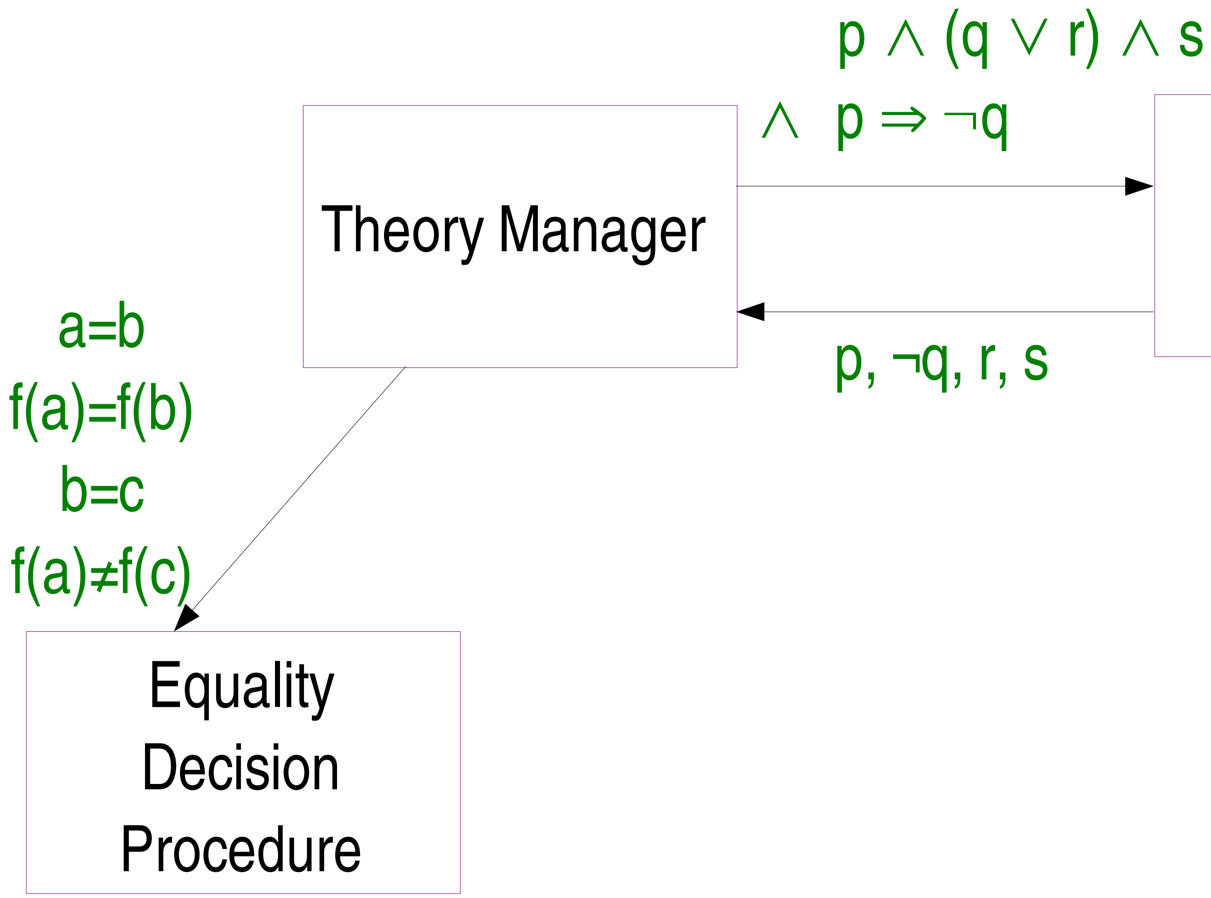
p: $a=b$

q: $f(a) \neq f(b)$

r: $b=c$

s: $f(a) \neq f(c)$

Example using lazy proof explication



<u>Mapping</u>

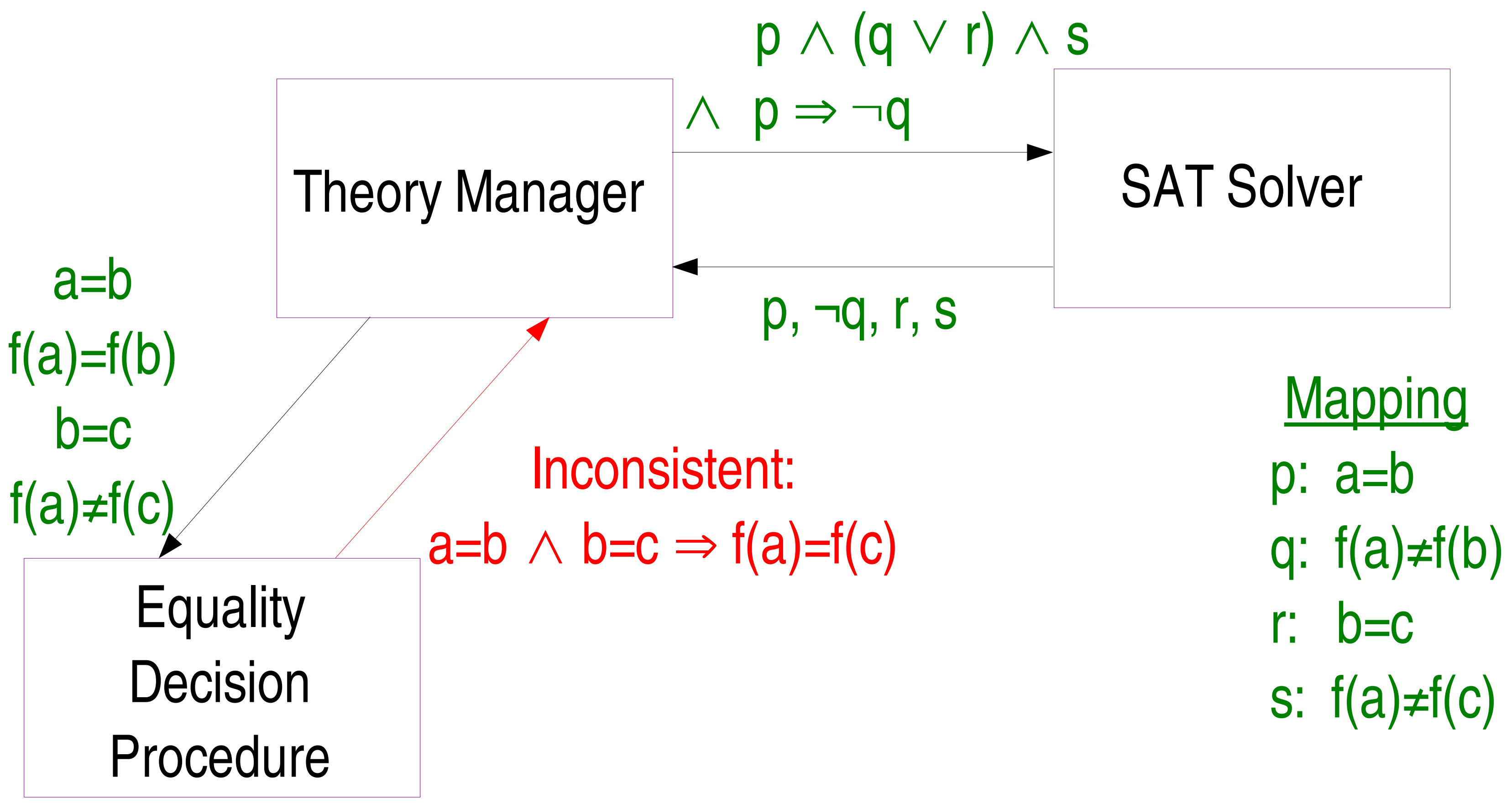
p: a=b

q: f(a)≠f(b)

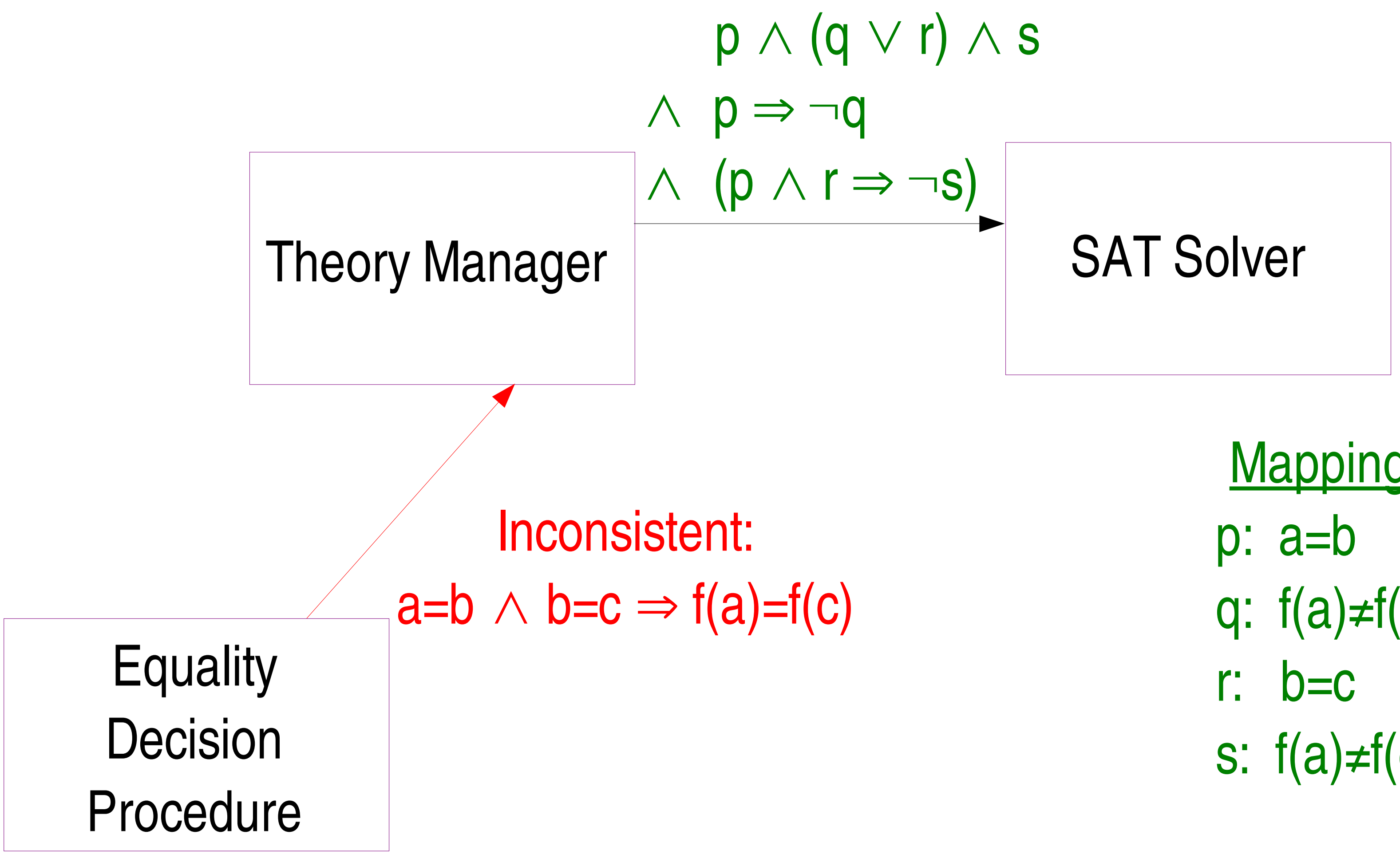
r: b=c

s: f(a)≠f(c)

Example using lazy proof explication



Example using lazy proof explication



<u>Mapping</u>

p: $a=b$

q: $f(a)\neq f(b)$

r: $b=c$

s: $f(a)\neq f(c)$

Example using lazy proof explication

$p \wedge (q \vee r) \wedge s$

$\wedge \;\; p \Rightarrow \neg q$

$\wedge \;\; (p \wedge r \Rightarrow \neg s)$

Theory Manager

SAT Solver

Unsatisfiable

Equality Decision Procedure

<u>Mapping</u>

p: $a=b$

q: $f(a) \neq f(b)$

r: $b=c$

s: $f(a) \neq f(c)$

Definitions

- A literal is an atomic formula or its negation, e.g, $(a<b)$
- A quantified formula is either a $\forall$-formula or its negation e.g., $\neg\forall y.F$ where F is a formula (we also write this as $\exists y.\neg F$)
- A formula is an arbitrary boolean combination of atomic formulae and quantified formulae, e.g, $(b > 0 \Rightarrow \forall x.(P(x) \vee \exists y.\neg Q(x,y)))$
- A monome is a set of literals and quantified formulae, e.g., $\{ b > 0, \neg Q(a,b), \forall x.(P(x) \vee \exists y.\neg Q(x,y)) \}$

Two key procedures

- *satisfyProp(F)*
  - returns either UNSAT, or
  - a monome *m* representing a satisfying boolean assignment to the atomic formulae and outermost quantified formulae in *F*
- *satisfyTheories(m)*
  - returns either SAT, or
  - a formula *F* such that

    F is a tautology wrt the underlying theories, and

    $(F \wedge m)$ is **propositionally** unsatisfiable

Algorithm for quantifier-free formulae

- satisfy(F) {   /* returns UNSAT or a monome satisfying F */

```
E := true
while (true) {
    m := satisfyProp(F ∧ E)
    if (m = UNSAT)  { return UNSAT }
    else {
        R := satisfyTheories(m)
        if (R = SAT) { return m }
        else { E := E ∧ R }
    }
}
```

Related Work

- CVC [Dill, Stump, Barrett], CVC-Lite [Barrett, Berezin]
- ICS [de Moura, Ruess, Shankar,]
- Math-SAT [Audemard, Bertoli, Cimatti, Kornilowicz, Sebastiani]
- DPLL(T) [Ganzinger, Hagen, Nieuwenhius,Oliveras,Tinelli]
- UCLID [Bryant, Velev, Strichman, Seshia, Lahiri]
- Zapato [Ball,Cook,Lahiri,Zhang]
- TSAT++ [Armando,Castellini,Giunchiglia,Idini,Maratea]

Further Information

- *Theorem Proving Using Lazy Proof Explication*
  Flanagan, Joshi, Ou, Saxe
  CAV 2003

- *An Explicating Theorem Prover for Quantified Formulas*
  Flanagan, Joshi, Saxe
  HP Tech Report (in preparation)