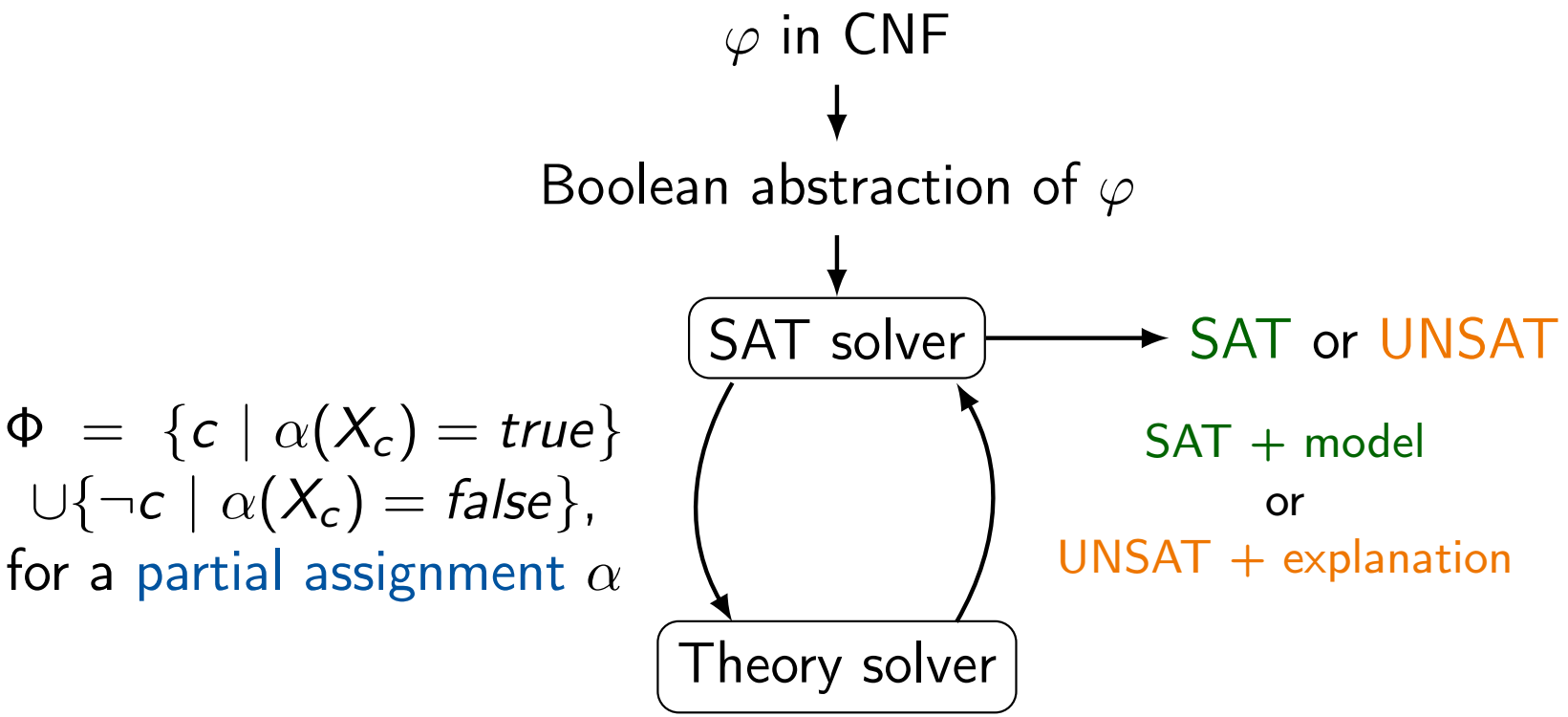
Practical Course: SMT Solving

Incrementality, Backtracking and Minimal Infeasible Subsets

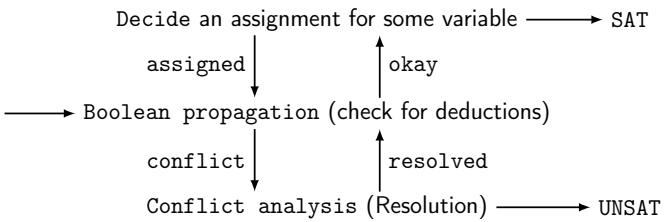
Erika Ábrahám, Rebecca Haehn, Gereon Kremer, Stefan Schupp

Winter term 2018/19

Reminder: Less Lazy SMT Solving + DPLL



Reminder: Less Lazy SMT Solving + DPLL



φ in CNF

Boolean abstraction of φ

SAT solver → SAT or UNSAT

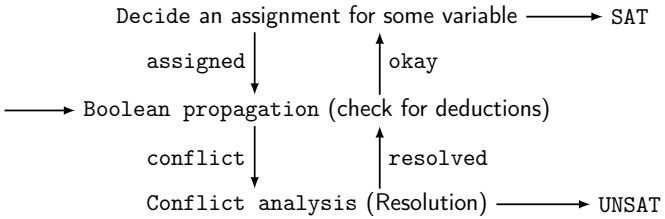
SAT + model
or
UNSAT + explanation

Theory solver

$$\Phi = \{c \mid \alpha(X_c) = \mathit{true}\} \cup \{\neg c \mid \alpha(X_c) = \mathit{false}\},$$

for a partial assignment α

Reminder: Less Lazy SMT Solving + DPLL



φ in CNF

Boolean abstraction of φ

SAT solver → SAT or UNSAT

$\Phi = \{c \mid \alpha(X_c) = \mathit{true}\} \cup \{\neg c \mid \alpha(X_c) = \mathit{false}\}$,
for a partial assignment α

SAT + model
or
UNSAT + explanation

Theory solver

→ many theory calls for similar sets of constraints

SMT-compliancy

1. **incrementality**: previous results should be reused and not recalculated if needed again (i.e. when the set of constraints becomes extended)

SMT-compliancy

1. **incrementality**: previous results should be reused and not recalculated if needed again (i.e. when the set of constraints becomes extended)
   - new equation is added: update partition and check previously added disequations for satisfiability
   - new disequation is added: check satisfiability of new disequation

SMT-compliancy

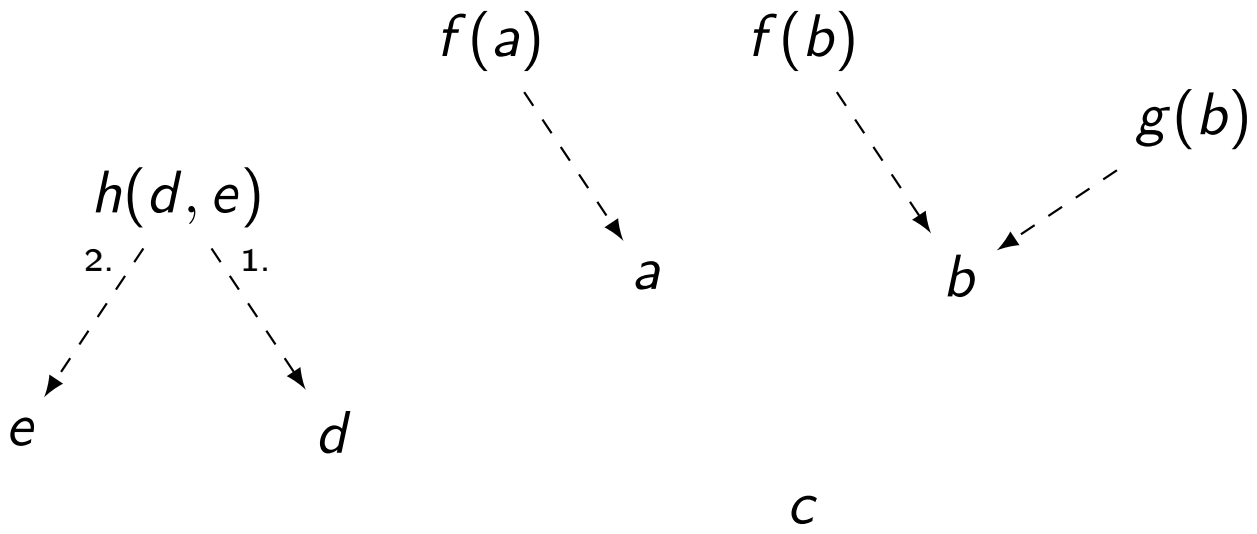
1. **incrementality**: previous results should be reused and not recalculated if needed again (i.e. when the set of constraints becomes extended)
   - new equation is added: update partition and check previously added disequations for satisfiability
   - new disequation is added: check satisfiability of new disequation
2. support **backtracking** according to the SAT solving (be able to remove constraints in inverse chronological order)

SMT-compliancy

1. **incrementality**: previous results should be reused and not recalculated if needed again (i.e. when the set of constraints becomes extended)
   - new equation is added: update partition and check previously added disequations for satisfiability
   - new disequation is added: check satisfiability of new disequation
2. support **backtracking** according to the SAT solving (be able to remove constraints in inverse chronological order)
   - remember computation history

Backtracking

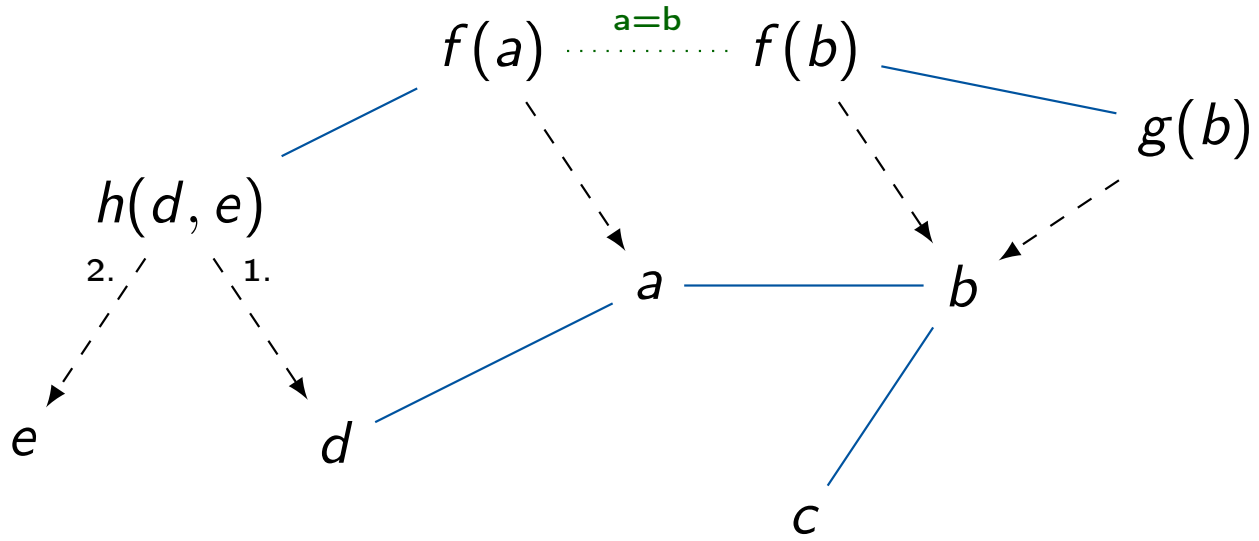
$\mathcal{T} = \{a,\ b,\ c,\ d,\ e,\ f(a),\ f(b),\ g(b),\ h(d,e)\}$
$\mathcal{C} = \{\}$



⇢ relations between terms

Backtracking

$$\mathcal{T} = \{a,\ b,\ c,\ d,\ e,\ f(a),\ f(b),\ g(b),\ h(d,e)\}$$
$$\mathcal{C} = \{a = d,\ b = c,\ f(a) = h(d,e),\ f(b) = g(b),\ a = b\}$$



relations between terms

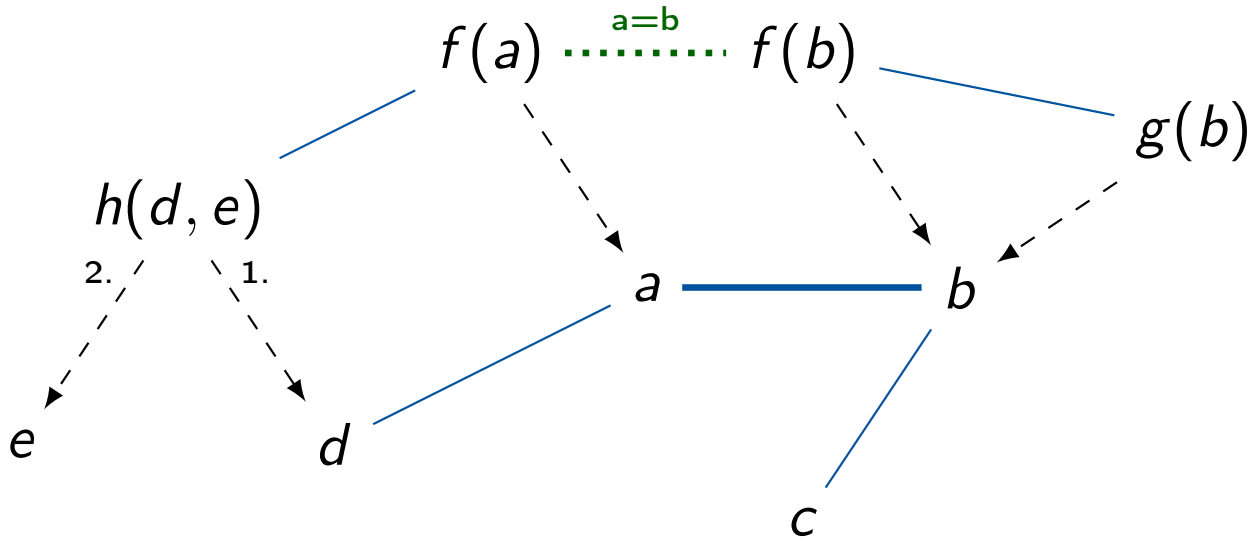
equalities

derived equalities

Backtracking

$\mathcal{T} = \{a,\ b,\ c,\ d,\ e,\ f(a),\ f(b),\ g(b),\ h(d,e)\}$
$\mathcal{C} = \{a = d,\ b = c,\ f(a) = h(d,e),\ f(b) = g(b)\}$

Remove $a = b$

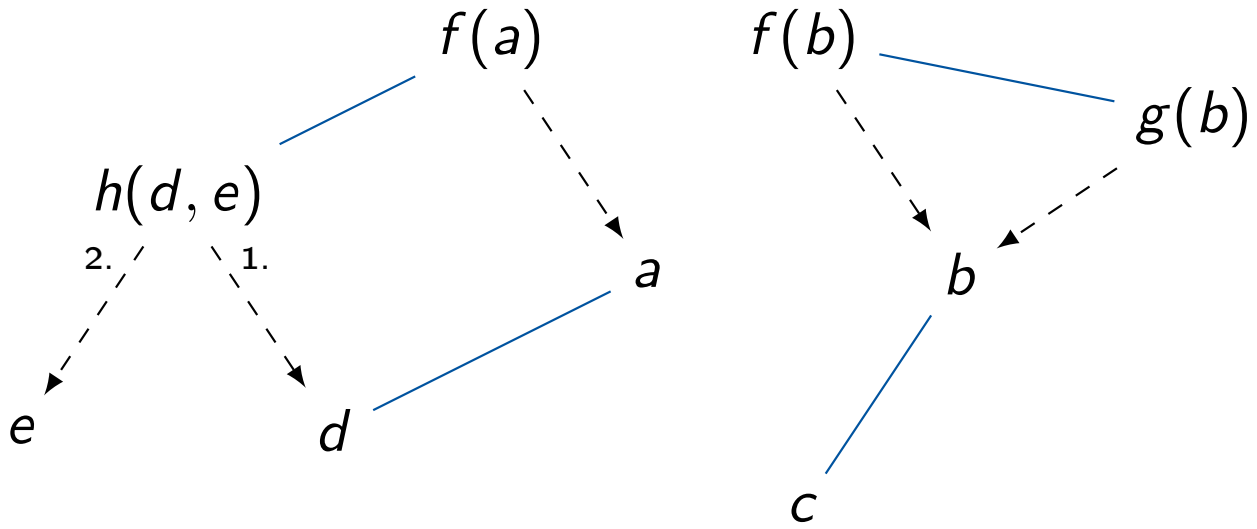


relations between terms
equalities
derived equalities

Backtracking

$\mathcal{T} = \{a,\ b,\ c,\ d,\ e,\ f(a),\ f(b),\ g(b),\ h(d,e)\}$
$\mathcal{C} = \{a = d,\ b = c,\ f(a) = h(d,e),\ f(b) = g(b)\}$

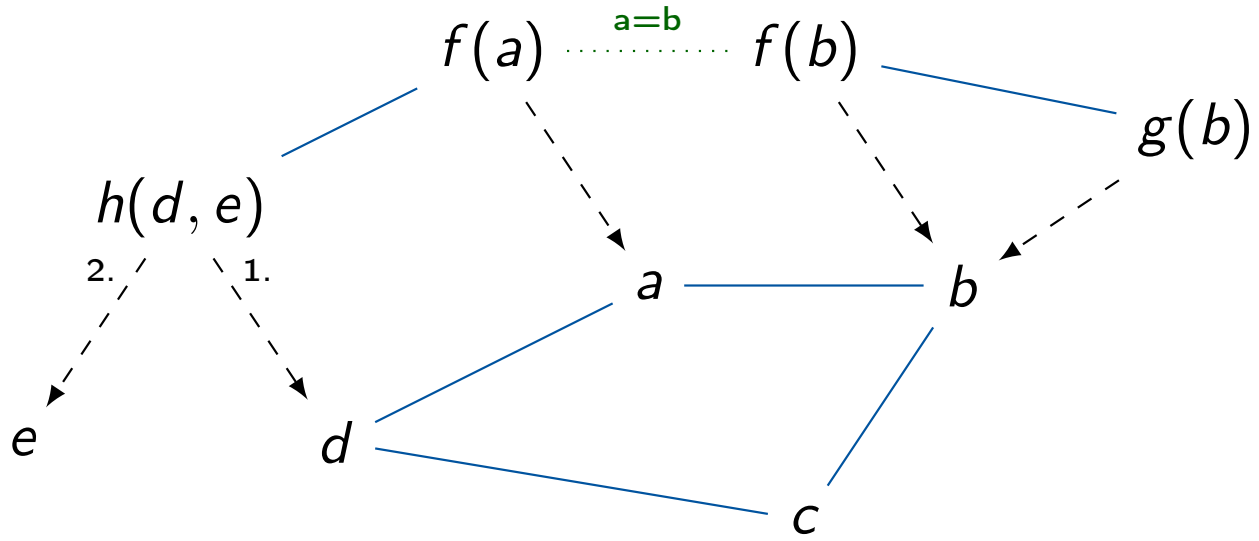
Removed $a = b$



relations between terms
equalities
derived equalities

Backtracking

$$\mathcal{T} = \{a,\ b,\ c,\ d,\ e,\ f(a),\ f(b),\ g(b),\ h(d,e)\}$$
$$\mathcal{C} = \{a = d,\ b = c,\ f(a) = h(d,e),\ f(b) = g(b),\ c = d,\ a = b\}$$

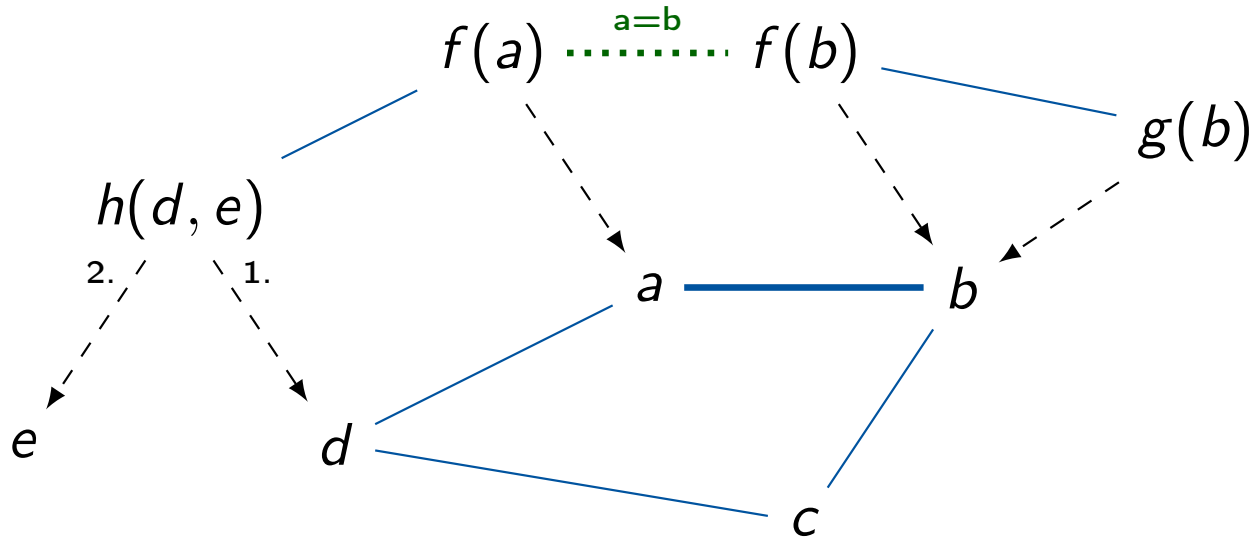


- - ▸ relations between terms
—— equalities
····· derived equalities

Backtracking

$\mathcal{T} = \{a,\ b,\ c,\ d,\ e,\ f(a),\ f(b),\ g(b),\ h(d,e)\}$
$\mathcal{C} = \{a = d,\ b = c,\ f(a) = h(d,e),\ f(b) = g(b),\ c = d\}$

Remove $a = b$

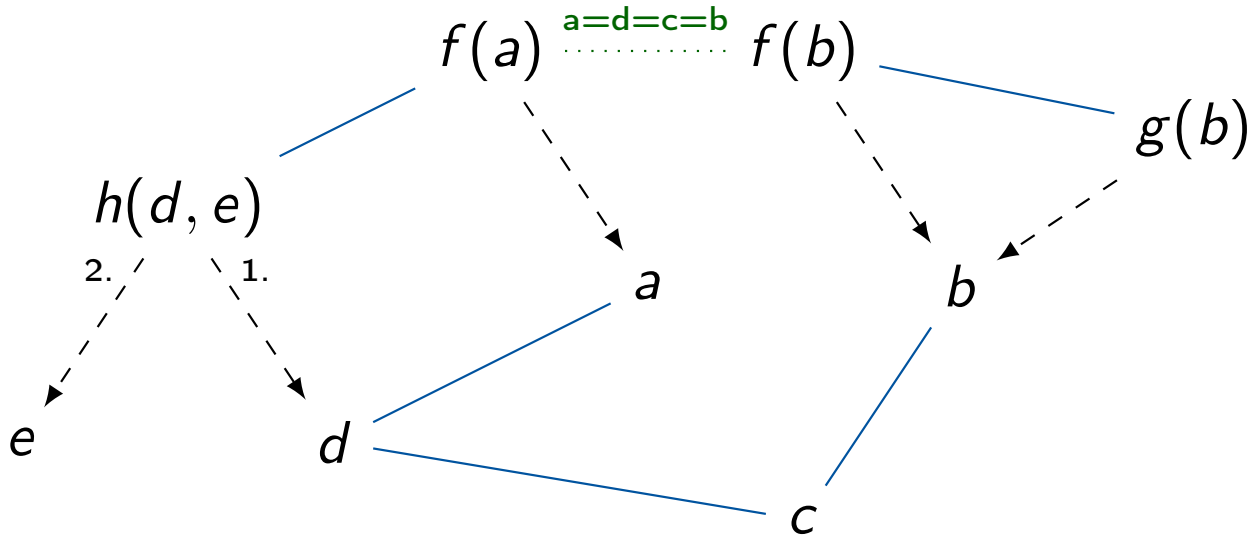


relations between terms
equalities
derived equalities

Backtracking

$\mathcal{T} = \{a,\ b,\ c,\ d,\ e,\ f(a),\ f(b),\ g(b),\ h(d,e)\}$
$\mathcal{C} = \{a = d,\ b = c,\ f(a) = h(d,e),\ f(b) = g(b),\ c = d\}$

Removed $a = b$



relations between terms
equalities
derived equalities

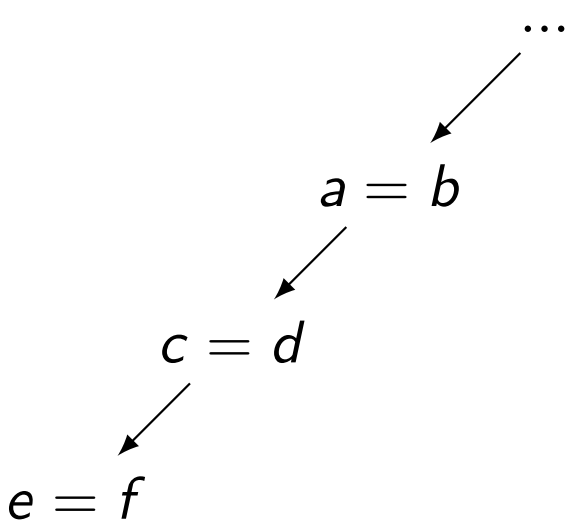
SMT-compliancy

1. **incrementality**: previous results should be reused and not recalculated if needed again (i.e. when the set of constraints becomes extended)
   - new equation is added: update partition and check previously added disequations for satisfiability
   - new disequation is added: check satisfiability of new disequation
2. support **backtracking** according to the SAT solving (be able to remove constraints in inverse chronological order)
   - remember computation history
3. finding a (preferably minimal) **infeasible subset** as explanation for inconsistent constraint sets

Infeasible Subsets

Why not just use the trivial infeasible subset?

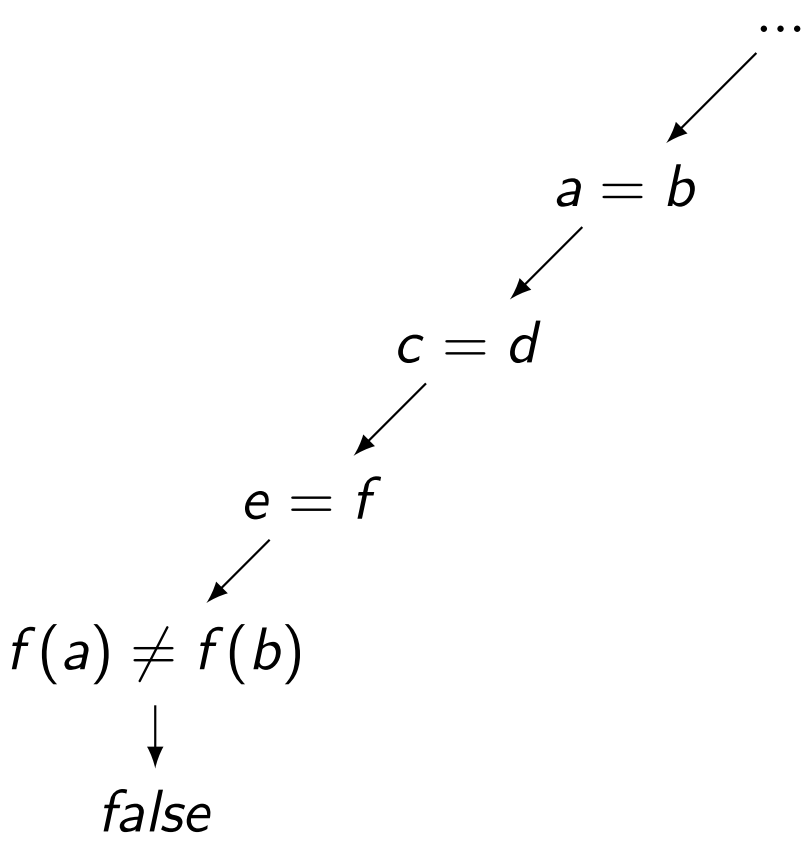
$$\ldots \wedge (\ldots \vee a = b) \wedge (e \neq f \vee f(a) \neq f(b)) \\ \wedge (c = d \vee f(a) \neq f(b)) \wedge (e = f \vee c \neq d \vee f(a) \neq f(b))$$



Infeasible Subsets

Why not just use the trivial infeasible subset?

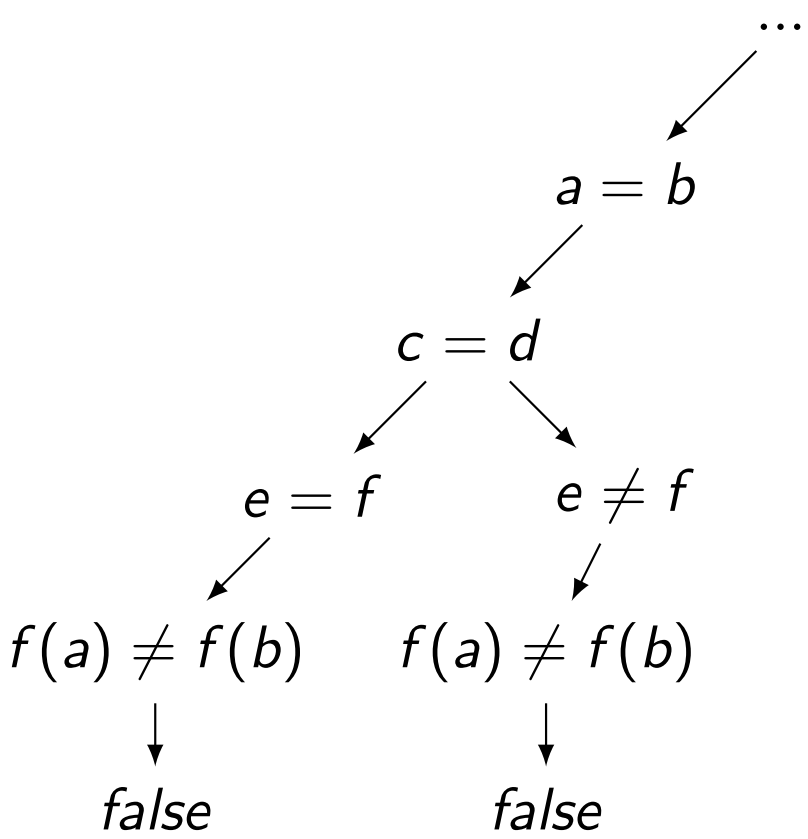
$$\ldots \wedge (\ldots \vee a = b) \wedge (e \neq f \vee f(a) \neq f(b)) \wedge (c = d \vee f(a) \neq f(b)) \wedge (e = f \vee c \neq d \vee f(a) \neq f(b))$$



Infeasible Subsets

Why not just use the trivial infeasible subset?

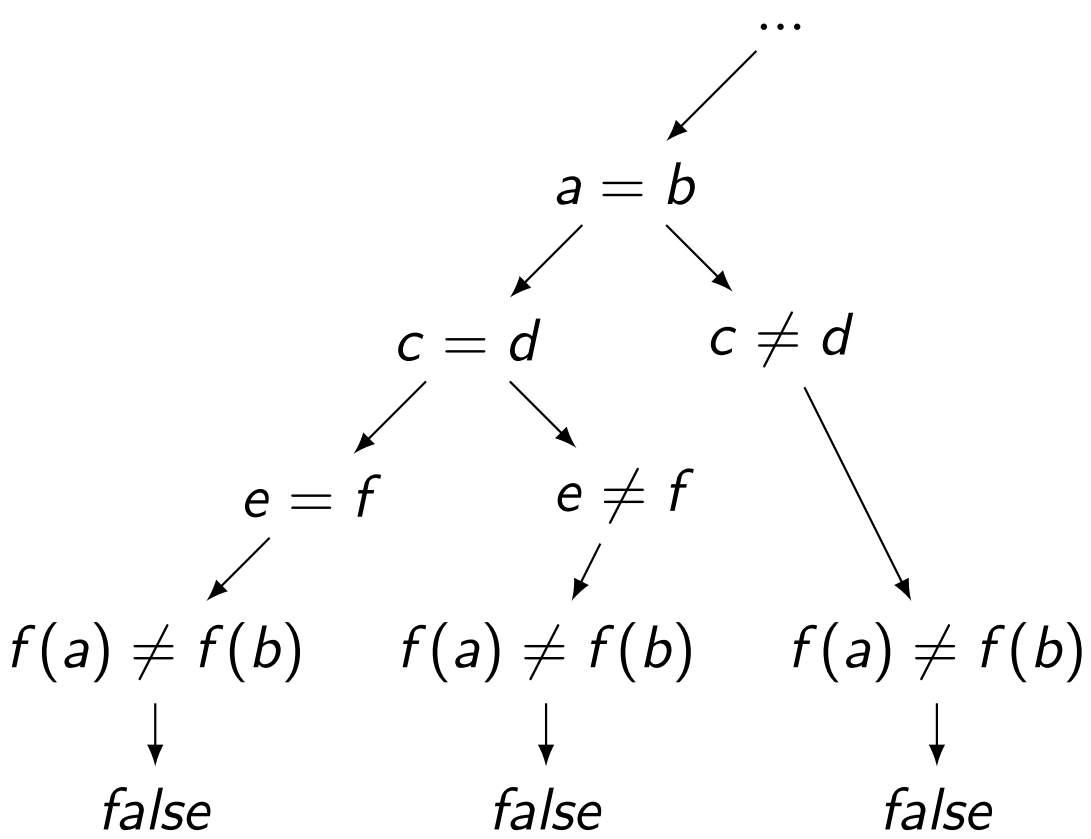
$$\ldots \wedge (\ldots \vee a = b) \wedge (e \neq f \vee f(a) \neq f(b))$$
$$\wedge (c = d \vee f(a) \neq f(b)) \wedge (e = f \vee c \neq d \vee f(a) \neq f(b))$$



Infeasible Subsets

Why not just use the trivial infeasible subset?

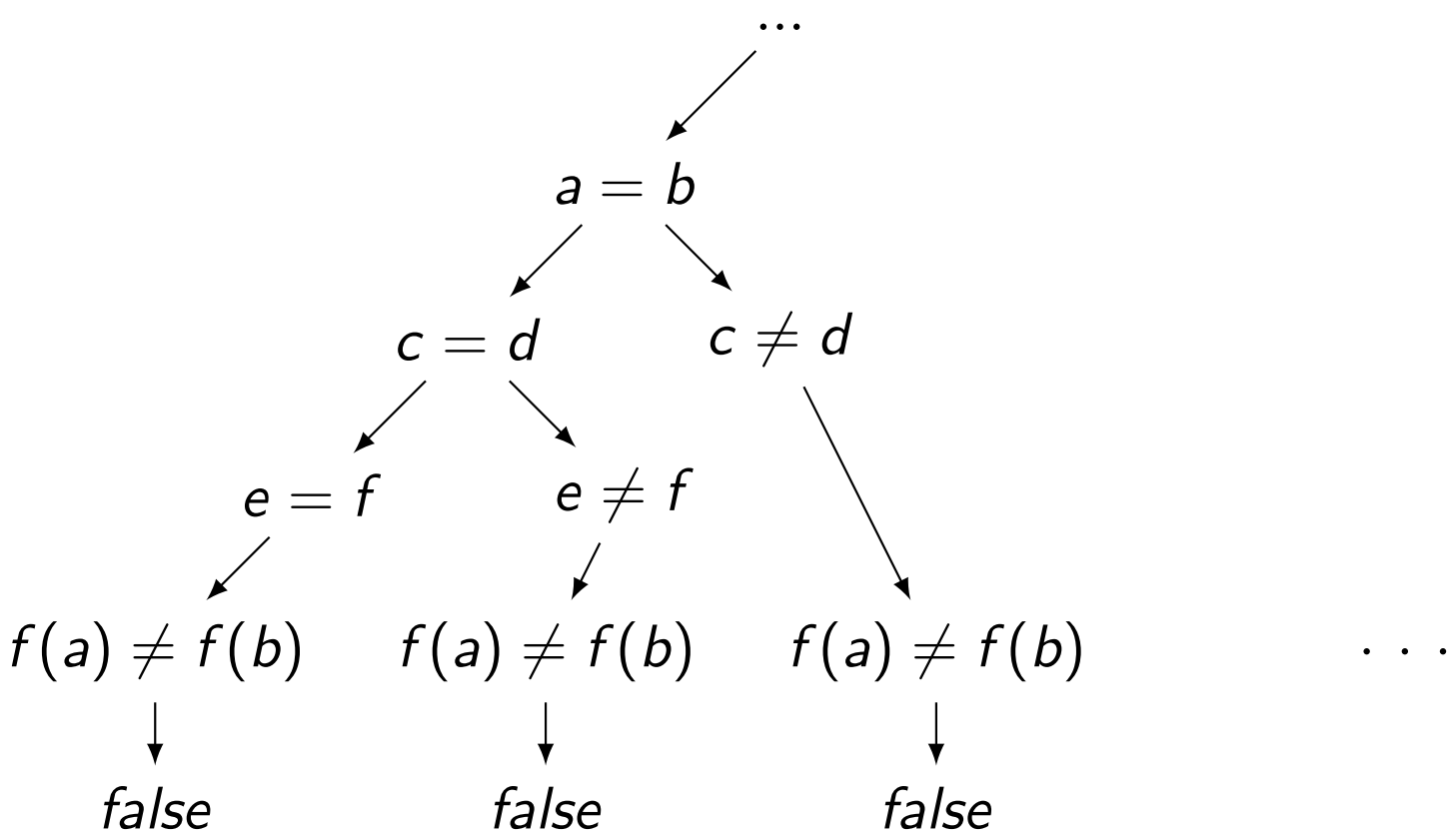
$$\ldots \wedge (\ldots \vee a = b) \wedge (e \neq f \vee f(a) \neq f(b))$$
$$\wedge (c = d \vee f(a) \neq f(b)) \wedge (e = f \vee c \neq d \vee f(a) \neq f(b))$$



Infeasible Subsets

Why not just use the trivial infeasible subset?

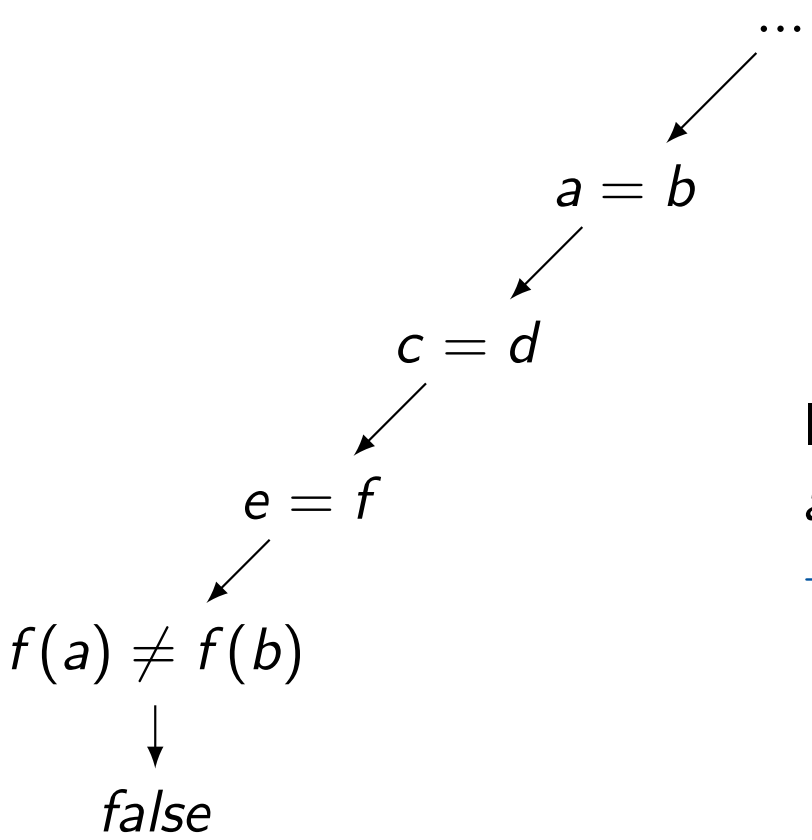
$$\ldots \wedge (\ldots \vee a = b) \wedge (e \neq f \vee f(a) \neq f(b)) \wedge (c = d \vee f(a) \neq f(b)) \wedge (e = f \vee c \neq d \vee f(a) \neq f(b))$$



Infeasible Subsets

When using the minimal infeasible subset:

$$\ldots \wedge (\ldots \vee a = b) \wedge (e \neq f \vee f(a) \neq f(b)) \wedge (c = d \vee f(a) \neq f(b)) \wedge (e = f \vee c \neq d \vee f(a) \neq f(b))$$

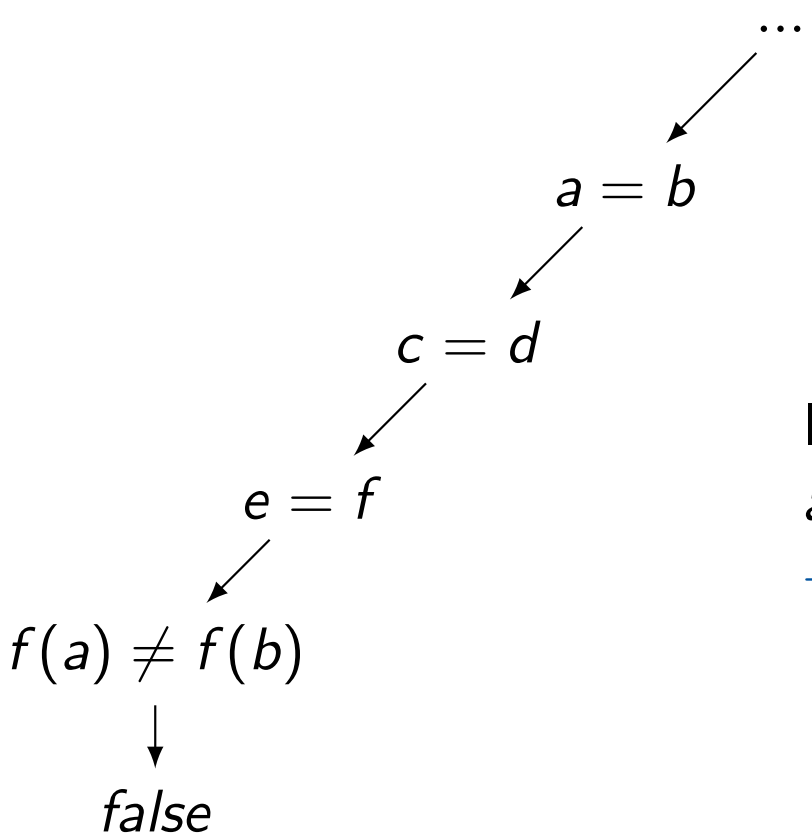


Infeasible subset:
$a = b \wedge f(a) \neq f(b)$
$\rightarrow$ Excludes more Boolean solutions

Infeasible Subsets

When using the minimal infeasible subset:

$$\ldots \wedge (\ldots \vee a = b) \wedge (e \neq f \vee f(a) \neq f(b)) \wedge (c = d \vee f(a) \neq f(b)) \wedge (e = f \vee c \neq d \vee f(a) \neq f(b))$$



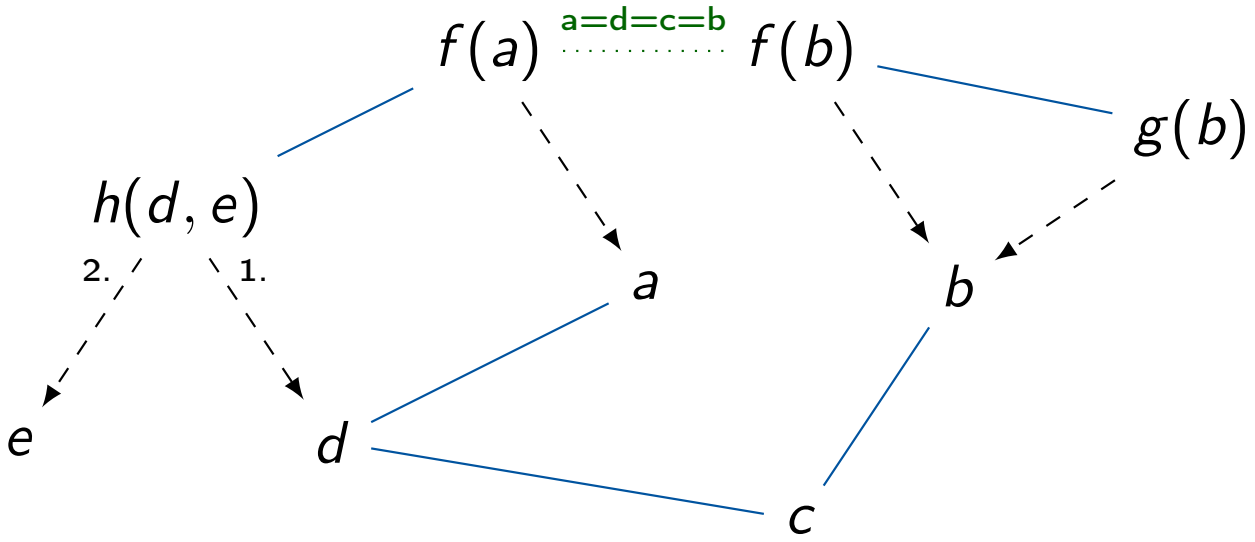
Infeasible subset:
$a = b \wedge f(a) \neq f(b)$
$\rightarrow$ Excludes more Boolean solutions

SMT-compliancy

1. **incrementality**: previous results should be reused and not recalculated if needed again (i.e. when the set of constraints becomes extended)
   - new equation is added: update partition and check previously added disequations for satisfiability
   - new disequation is added: check satisfiability of new disequation
2. support **backtracking** according to the SAT solving (be able to remove constraints in inverse chronological order)
   - remember computation history
3. finding a (preferably minimal) **infeasible subset** as explanation for inconsistent constraint sets
   - build set of $t \neq t'$ and (a minimal number of) equations that imply $t = t'$ by transitivity and congruence

Minimal Infeasible Subset

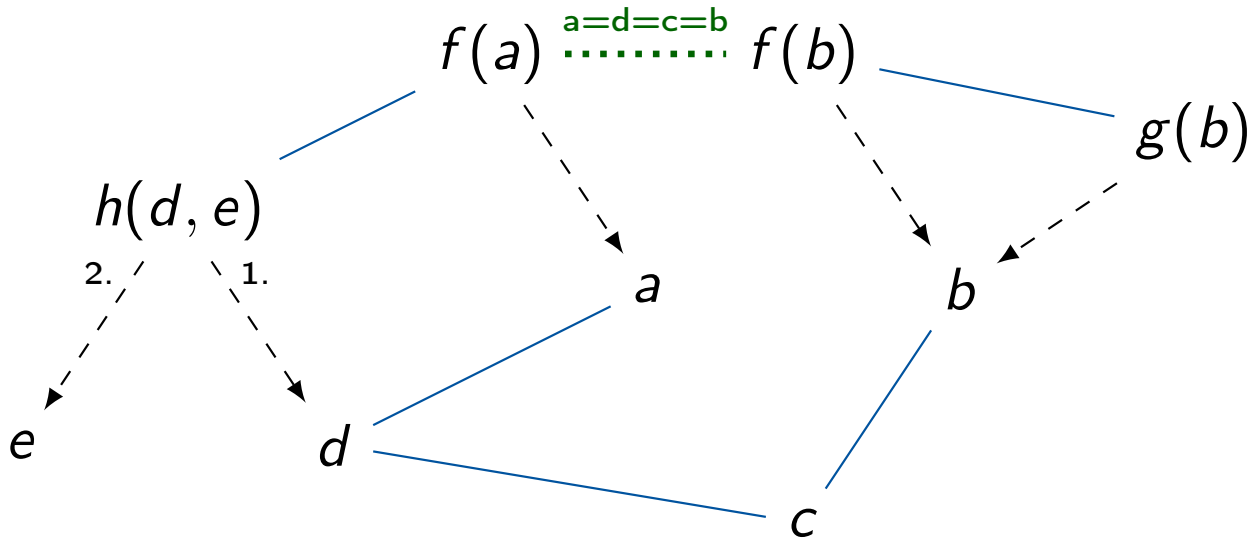
$$\mathcal{T} = \{a,\ b,\ c,\ d,\ e,\ f(a),\ f(b),\ g(b),\ h(d,e)\}$$
$$\mathcal{C} = \{a = d,\ b = c,\ f(a) = h(d,e),\ f(b) = g(b),\ c = d,\ f(a) \neq f(b)\}$$



Minimal Infeasible Subset

$\mathcal{T} = \{a,\ b,\ c,\ d,\ e,\ f(a),\ f(b),\ g(b),\ h(d,e)\}$
$\mathcal{C} = \{a = d,\ b = c,\ f(a) = h(d,e),\ f(b) = g(b),\ c = d,\ f(a) \neq f(b)\}$

Minimal infeasible subset: shortest 'path' between $f(a)$ and $f(b)$

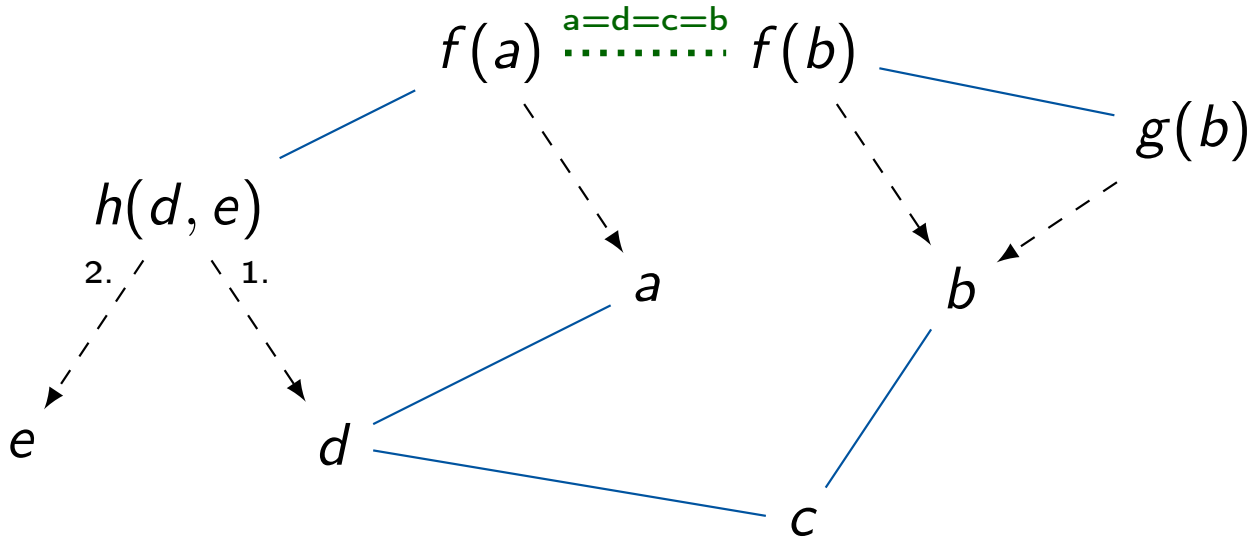


- $f(a) - f(b)$ is implied by congruence (equality of successors)

Minimal Infeasible Subset

$\mathcal{T} = \{a,\ b,\ c,\ d,\ e,\ f(a),\ f(b),\ g(b),\ h(d,e)\}$
$\mathcal{C} = \{a = d,\ b = c,\ f(a) = h(d,e),\ f(b) = g(b),\ c = d,\ f(a) \neq f(b)\}$

Minimal infeasible subset: shortest 'path' between $f(a)$ and $f(b)$

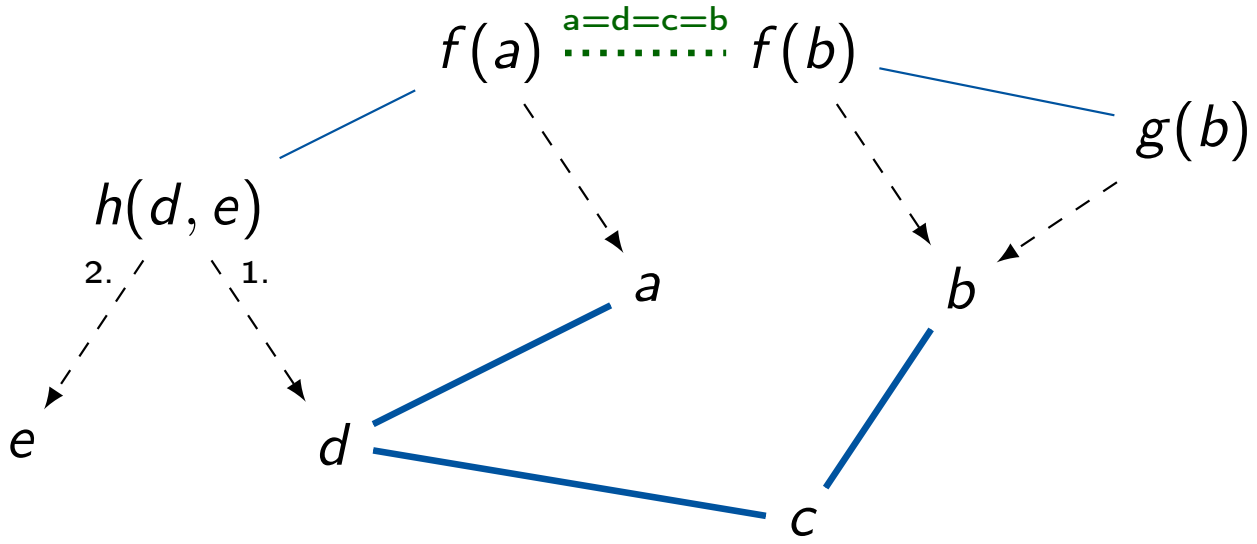


- $f(a) - f(b)$ is implied by congruence (equality of successors)
  $\rightarrow$ find shortest path between successors

Minimal Infeasible Subset

$$\mathcal{T} = \{a,\ b,\ c,\ d,\ e,\ f(a),\ f(b),\ g(b),\ h(d,e)\}$$
$$\mathcal{C} = \{a = d,\ b = c,\ f(a) = h(d,e),\ f(b) = g(b),\ c = d,\ f(a) \neq f(b)\}$$

Minimal infeasible subset: shortest 'path' between $f(a)$ and $f(b)$

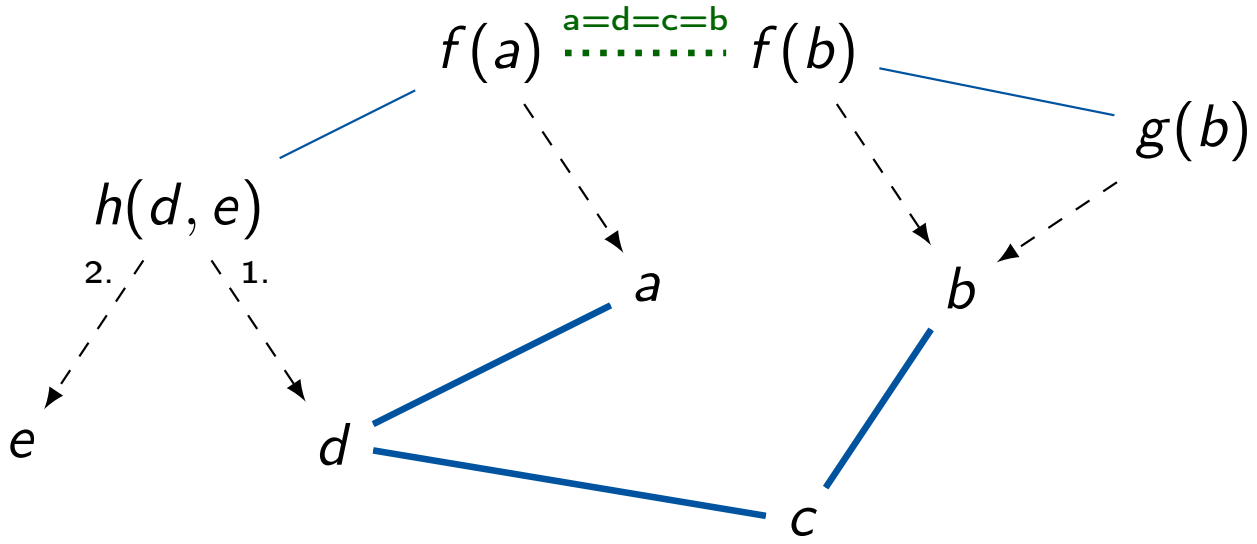


- $f(a) - f(b)$ is implied by congruence (equality of successors)
  $\rightarrow$ find shortest path between successors
- consists only of equality edges $\rightarrow$ done!

Minimal Infeasible Subset

$\mathcal{T} = \{a,\ b,\ c,\ d,\ e,\ f(a),\ f(b),\ g(b),\ h(d,e)\}$
$\mathcal{C} = \{a = d,\ b = c,\ f(a) = h(d,e),\ f(b) = g(b),\ c = d,\ f(a) \neq f(b)\}$

Minimal infeasible subset: shortest 'path' between $f(a)$ and $f(b)$



- $f(a) - f(b)$ is implied by congruence (equality of successors)
  $\rightarrow$ find shortest path between successors
- consists only of equality edges $\rightarrow$ done!
- Note that the resulting infeasible subset is not always minimal.

Theory Conflict

Algorithm: `getExplanation`

Input: State after some execution of `addEquation` or `addDisequation`
Output: If the current state is `UNSAT` an unsatisfiable subset MIS of the previously received equations and disequations else the empty set

1. $MIS := \emptyset$; `if (state = SAT) then return` MIS;
2. find a conflicting disequation: $t_1 \neq t_2$
3. find the shortest path P in the E-graph between the nodes t_1 and t_2
4. `while` $P \neq \emptyset$ `let` $e = (t, t') \in P$
   `if` $t = t' \in E$ `then` $MIS := MIS \cup \{t = t'\}$;
   `else let` $t = f(x_1, ..., x_n)$, $t' = f(y_1, ..., y_n)$ `for some` x, y `and` f
   `for` $k = 1$, $k < n$, $k{+}{+}$
   `find the shortest path` P_k `between` x_k `and` y_k
   $P := P \cup P_k$;

`return` MIS;