How Does an Answer Set Solver Work?

Notes

- Assume that *P* is a ground and normal program.
- Answer sets of disjunctive programs are computed in a similar way.
- The algorithm presented in these slides is similar to the algorithm implemented by the first answer set solver (smodels)

Useful Observations about Answer Sets

- For a program *P* and an answer set *S*
  - if an atom *a* does not appear in the head of any rule in *P* then $a \notin S$.
  - if an atom $a \in S$ then there exists a rule *r* in *P* such that $a = head(r)$, $pos(r) \subseteq S$, and $neg(r) \cap S = \emptyset$.
- An answer set S can be viewed as a pair <S,N> where $N = B_P \setminus S$ (B_P is the Herbrand base of P; S contains the true atoms and N contains the false atoms)

Useful Observations about Answer Sets

- If we know that some atom, say a, must be in an answer set then we could eliminate all the rules, whose negative part contains a, from consideration
- If we know that some atom, say a, cannot be true in an answer set then we could eliminate all the rules, whose positive part contains a, from further consideration

P_1:

$a \leftarrow$

$b \leftarrow \text{not } a$

$c \leftarrow b$

a must be in every answer set (1)

in virtue of (1) this rule should never be considered or b should be false (2)

in virtue of (2) this rule can never be applied and hence c cannot be true (3)

(1), (2), (3) imply that the answer set is {a}

Useful Observations about Answer Sets

- A partial answer set is a pair <CS, CN> where CS and CN are two disjoint sets of atoms in B_P which contain atoms that must be true or false, respectively, with respect to an answer set
- Given P and a partial answer set <CS, CN> the aforementioned observations can be used to determine a partial answer set <CS', CN'> such that CS $\subseteq$ CS' and CN $\subseteq$ CN' if CS is a subset of an answer set
  - E.g.: for P_1 and <$\emptyset$, $\emptyset$> leads to <{a},{b}>

Useful Observations about Answer Sets

- Given P and a partial answer set <CS, CN>, there are situations where no atom must be true or false. In this case, the value of an atom must be guessed and depending on the guessed value, the partial answer set can become different. E.g.,

  $P_2 = \{a \leftarrow \text{not } b \text{ , } b \leftarrow \text{not } a\}$ and $<\emptyset, \emptyset>$

  a could be true or false and b could be true or false.

  Guessing a true leads to $<\{a\}, \emptyset>$ which ultimately leads to $<\{a\}, \{b\}>$

  Guessing a false leads to $<\emptyset, \{a\}>$ which ultimately leads to $<\{b\}, \{a\}>$

Detailed Algorithm: Expand(P, CS, CN)

- Input: a program P and a partial answer set <CS,CN>
- Output: a partial answer set <CS', CN'> such that CS $\subseteq$CS' and CN $\subseteq$CN' or **false** if CS cannot be extended to an answer set

repeat

- set *change* to **false**
- find all rule r such that pos(r)$\subseteq$CS and neg(r)$\subseteq$CN, add head(r) to CS, and set *change* to **true**
- find all rule r such that pos(r)$\cap$CN$\neq\emptyset$ or neg(r)$\cap$CS$\neq\emptyset$, add head(r) to CN, and set *change* to **true**

until there is no change in <CS,CN> (*change* is **false**)

return <CS,CN> if CS $\cap$CN=$\emptyset$ or **false** otherwise

Detailed Algorithm: Solves(P, CS, CN)

- Input: a program P, a partial answer set <CS, CN>
- Output: answer sets of P which are superset of CS or false otherwise

if Expand(P, CS, CN) = **false then return false**

<CS,CN> = Expand(P, CS, CN)

select an atom a that does not belong to CS $\cup$CN

return Solves(P,CS$\cup${a},CN)$\cup$Solves(P,CS,CN$\cup${a})

Solver Algorithm

- Input: a program P
- Output: answer sets of P or false if no answer set exists

if Expand(P, $\emptyset$, $\emptyset$) = **false** then **return false**

set <CS,CN> = Expand(P, $\emptyset$, $\emptyset$)

return Solves(P, CS, CN)

Example

P = {
- a ←
- b ← not a
- c ← a, d
- e ← not d
- d ← not e

Which atom to guess is the key to solver's performance

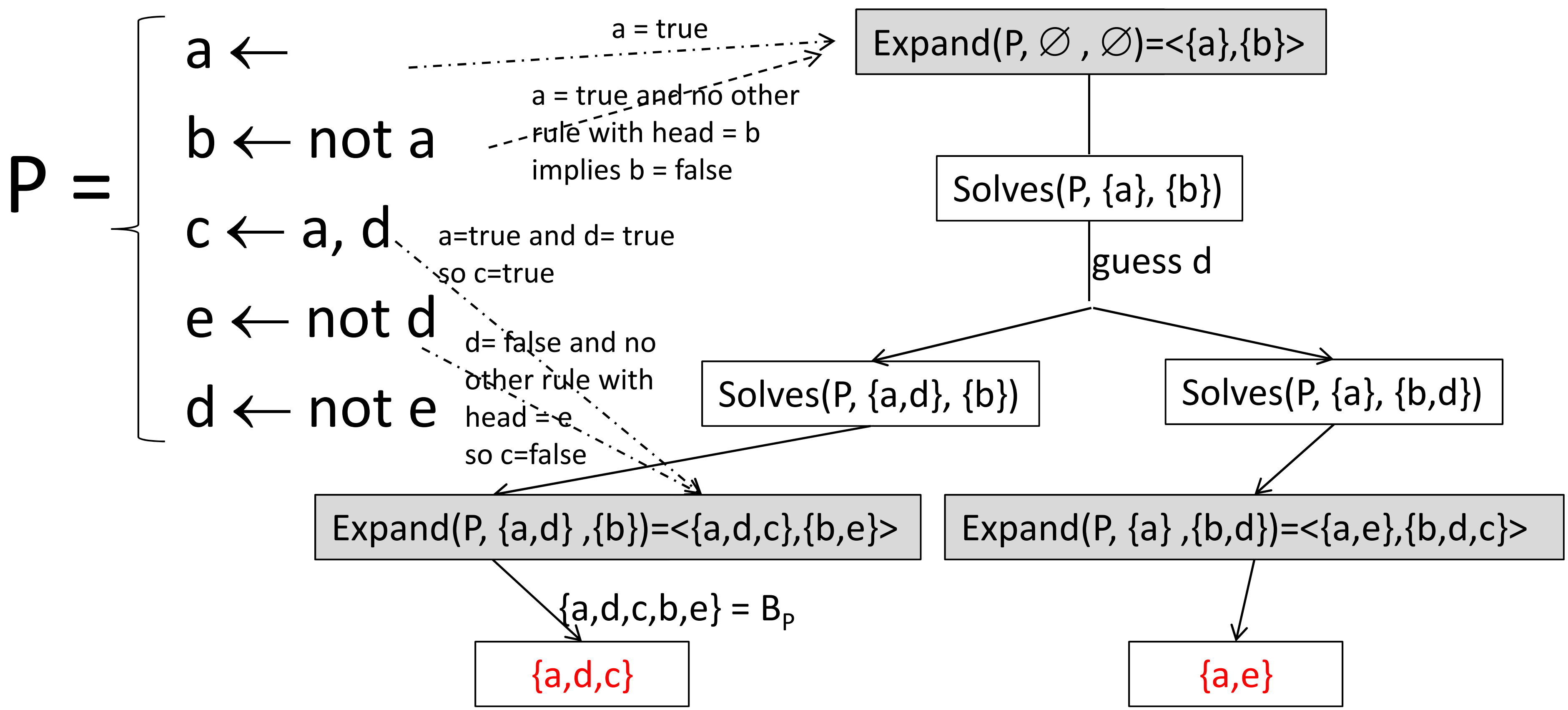
Expand(P, ∅ , ∅) returns $\langle \{a\}, \{b\} \rangle$

Solves(P, {a}, {b}) calls Solves(P, {a,d}, {b})
and Solves(P, {a}, {b,d}) [Guessing d]

Solves(P, {a,d}, {b}) returns {a,d,c} as an answer set

Solves(P, {a}, {b,d}) returns {a,e} as an answer set

Example



P =
a ←
b ← not a
c ← a, d
e ← not d
d ← not e
a = true
a = true and no other rule with head = b implies b = false
a=true and d= true so c=true
d= false and no other rule with head = e so c=false
Expand(P, ∅ , ∅)=<{a},{b}>
Solves(P, {a}, {b})
guess d
Solves(P, {a,d}, {b})
Solves(P, {a}, {b,d})
Expand(P, {a,d} ,{b})=<{a,d,c},{b,e}>
Expand(P, {a} ,{b,d})=<{a,e},{b,d,c}>
{a,d,c,b,e} = B_P
{a,d,c}
{a,e}