Informatics 2D · Agents and Reasoning · 2019/2020

# Lecture 12 · Resolution-Based Inference

Claudia Chirita

School of Informatics, University of Edinburgh

7th February 2020

Based on slides by: Jacques Fleuriot, Michael Rovatsos, Michael Herrmann, Vaishak Belle

# Previously on INF2D

## The Hundred-Acre Wood

$\text{VeryFondOfFood}(x) \wedge \text{Treat}(y) \wedge \text{Friend}(z) \wedge \text{Gives}(x, y, z) \rightarrow \text{Generous}(x)$

$\exists x.\text{Owns}(\text{Eeyore}, x) \wedge \text{Hunny}(x)$ $\quad$ $\text{Owns}(\text{Eeyore}, \text{J}) \wedge \text{Hunny}(\text{J})$

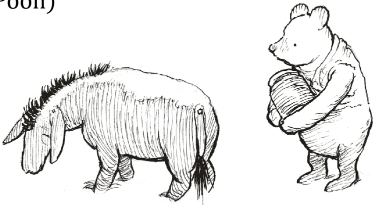
$\text{Hunny}(x) \wedge \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

$\text{Hunny}(x) \rightarrow \text{Treat}(x)$

$\text{Resident}(x, \text{HundredAcreWood}) \rightarrow \text{Friend}(x)$

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

$\text{VeryFondOfFood}(\text{Pooh})$



# Outline

- Forward chaining
- Backward chaining
- Resolution

# Forward chaining algorithm

**function** FOL-FC-ASK($KB, \alpha$) **returns** a substitution or *false*

  **inputs**: $KB$, the knowledge base, a set of first-order definite clauses

    $\alpha$, the query, an atomic sentence

  **local variables**: $new$, the new sentences inferred on each iteration

  **repeat until** $new$ is empty

    $new \leftarrow \{\,\}$

    **for each** $rule$ in $KB$ **do**

      $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-VARIABLES($rule$)

*Replaces all variables in its arguments with new ones*

      **for each** $\theta$ such that $\text{SUBST}(\theta, p_1 \wedge \ldots \wedge p_n) = \text{SUBST}(\theta, p'_1 \wedge \ldots \wedge p'_n)$

        for some $p'_1, \ldots, p'_n$ in $KB$

*Pattern-matching*

        $q' \leftarrow \text{SUBST}(\theta, q)$

        **if** $q'$ does not unify with some sentence already in $KB$ or $new$ **then**

          add $q'$ to $new$

*Facts irrelevant to the goal can be generated*

          $\phi \leftarrow \text{UNIFY}(q', \alpha)$

          **if** $\phi$ is not $fail$ **then return** $\phi$

    add $new$ to $KB$

  **return** *false*

# Example · Forward chaining proof

$\text{VeryFondOfFood}(x) \land \text{Treat}(y) \land \text{Friend}(z) \land \text{Gives}(x, y, z) \rightarrow \text{Generous}(x)$

$\text{Owns}(\text{Eeyore}, \text{J}) \land \text{Hunny}(\text{J})$

$\text{Hunny}(x) \land \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

$\text{Hunny}(x) \rightarrow \text{Treat}(x)$

$\text{Resident}(x, \text{HundredAcreWood}) \rightarrow \text{Friend}(x)$

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

$\text{VeryFondOfFood}(\text{Pooh})$

VeryFondOfFood(Pooh) | Hunny(J) | Owns(Eeyore,J) | Resident(Eeyore,HAW)

# Example · Forward chaining proof

$\text{VeryFondOfFood}(x) \land \text{Treat}(y) \land \text{Friend}(z) \land \text{Gives}(x, y, z) \rightarrow \text{Generous}(x)$

$\text{Owns}(\text{Eeyore}, \text{J}) \land \text{Hunny}(\text{J})$

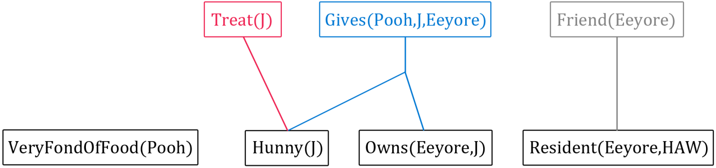
$\text{Hunny}(x) \land \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

$\text{Hunny}(x) \rightarrow \text{Treat}(x)$

$\text{Resident}(x, \text{HundredAcreWood}) \rightarrow \text{Friend}(x)$

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

$\text{VeryFondOfFood}(\text{Pooh})$



# Example · Forward chaining proof

$\text{VeryFondOfFood}(x) \wedge \text{Treat}(y) \wedge \text{Friend}(z) \wedge \text{Gives}(x, y, z) \rightarrow \text{Generous}(x)$

$\text{Owns}(\text{Eeyore}, \text{J}) \wedge \text{Hunny}(\text{J})$

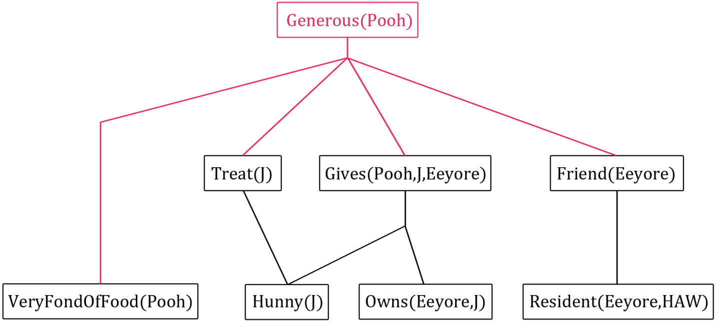
$\text{Hunny}(x) \wedge \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

$\text{Hunny}(x) \rightarrow \text{Treat}(x)$

$\text{Resident}(x, \text{HundredAcreWood}) \rightarrow \text{Friend}(x)$

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

$\text{VeryFondOfFood}(\text{Pooh})$



# Properties of forward chaining

FC is sound and complete for first-order definite clauses (exactly one positive literal).

Datalog = first-order definite clauses + no functions.
FC terminates for Datalog in a finite number of iterations.

May not terminate in general if the query $q$ is not entailed.
This is unavoidable: entailment with definite clauses is semi-decidable.

# Efficiency of forward chaining

Incremental forward chaining: no need to match a rule on iteration $k$ if a premise wasn't added on iteration $k-1$
$\Rightarrow$ match each rule whose premise contains a newly added positive literal.

Matching itself can be expensive:
Database indexing allows $O(1)$ retrieval of known facts.
e.g. query $\text{Hunny}(x)$ retrieves $\text{Hunny}(J)$

Forward chaining is widely used in deductive databases.

## Pattern Matching

- for each $\theta$ s.t. $\text{SUBST}(\theta, p_1 \wedge \ldots \wedge p_n) = \text{SUBST}(\theta, p'_1 \wedge \ldots \wedge p'_n)$ for some $p'_1, \ldots, p'_n$ in KB
- Finding all possible unifiers can be very expensive.

# Efficiency of forward chaining

Example

$\text{Hunny}(x) \land \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

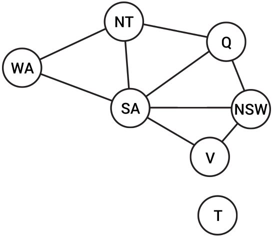
Can find each object owned by Eeyore in constant time and then check if it is a jar of hunny.

But what if Eeyore owns many objects but very few jars?

Conjunct Ordering: Better (cost-wise) to find all jars of hunny first and then check whether they are owned by Eeyore.

Optimal ordering is NP-hard. Heuristics available: MRV from CSP if each conjunct is viewed as a constraint on its variables.

# Hard matching example




$$\text{Diff}(WA, NT) \wedge \text{Diff}(WA, SA) \wedge \text{Diff}(NT, Q) \wedge$$
$$\text{Diff}(NT, SA) \wedge \text{Diff}(Q, NSW) \wedge \text{Diff}(Q, SA) \wedge$$
$$\text{Diff}(NSW, V) \wedge \text{Diff}(NSW, SA) \wedge$$
$$\text{Diff}(V, SA) \rightarrow \text{Colourable}$$

$$\text{Diff}(\text{Red}, \text{Blue}),\ \text{Diff}(\text{Red}, \text{Black})$$
$$\text{Diff}(\text{Black}, \text{Red}),\ \text{Diff}(\text{Black}, \text{Blue})$$
$$\text{Diff}(\text{Blue}, \text{Red}),\ \text{Diff}(\text{Blue}, \text{Black})$$

Every finite domain CSP can be expressed as a single definite clause + ground facts.

Colourable is inferred iff the CSP has a solution.

CSPs include 3SAT as a special case, hence matching is NP-hard.

# Backward chaining algorithm

A function that returns multiple times, each time giving one possible result

```
function FOL-BC-ASK(KB, query) returns a generator of substitutions
  return FOL-BC-OR(KB, query, { })
```

Fetch rules that might unify

```
generator FOL-BC-OR(KB, goal, θ) yields a substitution
  for each rule (lhs ⇒ rhs) in FETCH-RULES-FOR-GOAL(KB, goal) do
    (lhs, rhs) ← STANDARDIZE-VARIABLES((lhs, rhs))
    for each θ′ in FOL-BC-AND(KB, lhs, UNIFY(rhs, goal, θ)) do
      yield θ′
```

Renaming of variables to avoid name clashes

```
generator FOL-BC-AND(KB, goals, θ) yields a substitution
  if θ = failure then return
  else if LENGTH(goals) = 0 then yield θ
  else do
    first,rest ← FIRST(goals), REST(goals)
    for each θ′ in FOL-BC-OR(KB, SUBST(θ, first), θ) do
      for each θ″ in FOL-BC-AND(KB, rest, θ′) do
        yield θ″
```

# Example · Backward chaining

$\text{VeryFondOfFood}(x) \wedge \text{Treat}(y) \wedge \text{Friend}(z) \wedge \text{Gives}(x, y, z) \rightarrow \text{Generous}(x)$

$\text{Owns}(\text{Eeyore}, \text{J})$ and $\text{Hunny}(\text{J})$

$\text{Hunny}(x) \wedge \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

$\text{Hunny}(x) \rightarrow \text{Treat}(x)$

$\text{Resident}(x, \text{HundredAcreWood}) \rightarrow \text{Friend}(x)$

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

$\text{VeryFondOfFood}(\text{Pooh})$

$\boxed{\text{Generous}(\text{Pooh})}$

# Example · Backward chaining

$\text{VeryFondOfFood}(x) \land \text{Treat}(y) \land \text{Friend}(z) \land \text{Gives}(x, y, z) \rightarrow \text{Generous}(x)$

$\text{Owns}(\text{Eeyore}, \text{J})$ and $\text{Hunny}(\text{J})$

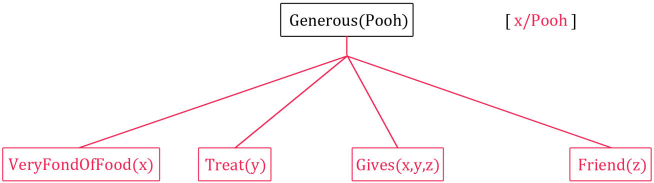
$\text{Hunny}(x) \land \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

$\text{Hunny}(x) \rightarrow \text{Treat}(x)$

$\text{Resident}(x, \text{HundredAcreWood}) \rightarrow \text{Friend}(x)$

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

$\text{VeryFondOfFood}(\text{Pooh})$



# Example · Backward chaining

$\text{VeryFondOfFood}(x) \land \text{Treat}(y) \land \text{Friend}(z) \land \text{Gives}(x, y, z) \rightarrow \text{Generous}(x)$

$\text{Owns}(\text{Eeyore}, \text{J})$ and $\text{Hunny}(\text{J})$

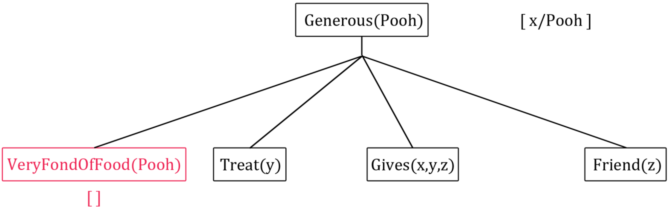
$\text{Hunny}(x) \land \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

$\text{Hunny}(x) \rightarrow \text{Treat}(x)$

$\text{Resident}(x, \text{HundredAcreWood}) \rightarrow \text{Friend}(x)$

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

$\text{VeryFondOfFood}(\text{Pooh})$



# Example · Backward chaining

$\text{VeryFondOfFood}(x) \land \text{Treat}(y) \land \text{Friend}(z) \land \text{Gives}(x, y, z) \rightarrow \text{Generous}(x)$

$\text{Owns}(\text{Eeyore}, \text{J})$ and $\text{Hunny}(\text{J})$

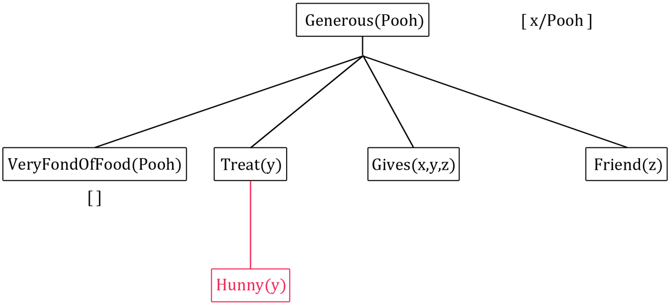
$\text{Hunny}(x) \land \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

$\text{Hunny}(x) \rightarrow \text{Treat}(x)$

$\text{Resident}(x, \text{HundredAcreWood}) \rightarrow \text{Friend}(x)$

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

$\text{VeryFondOfFood}(\text{Pooh})$



# Example · Backward chaining

$\text{VeryFondOfFood}(x) \wedge \text{Treat}(y) \wedge \text{Friend}(z) \wedge \text{Gives}(x, y, z) \rightarrow \text{Generous}(x)$

$\text{Owns}(\text{Eeyore}, \text{J})$ and $\text{Hunny}(\text{J})$

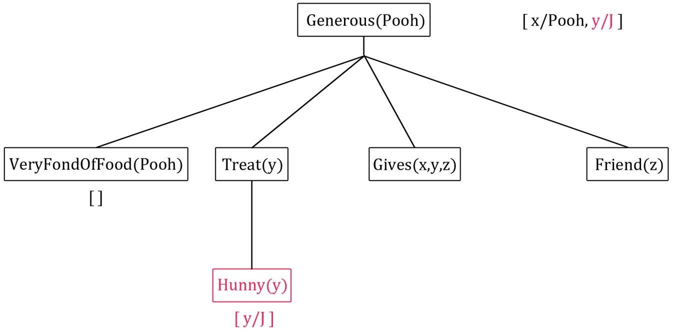
$\text{Hunny}(x) \wedge \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

$\text{Hunny}(x) \rightarrow \text{Treat}(x)$

$\text{Resident}(x, \text{HundredAcreWood}) \rightarrow \text{Friend}(x)$

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

$\text{VeryFondOfFood}(\text{Pooh})$



# Example · Backward chaining

VeryFondOfFood($x$) ∧ Treat($y$) ∧ Friend($z$) ∧ Gives($x, y, z$) → Generous($x$)

Owns(Eeyore, J) and Hunny(J)

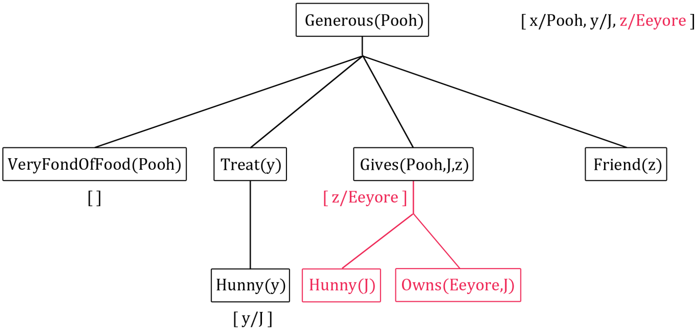
Hunny($x$) ∧ Owns(Eeyore, $x$) → Gives(Pooh, $x$, Eeyore)

Hunny($x$) → Treat($x$)

Resident($x$, HundredAcreWood) → Friend($x$)

Resident(Eeyore, HundredAcreWood)

VeryFondOfFood(Pooh)



# Example · Backward chaining

$\text{VeryFondOfFood}(x) \land \text{Treat}(y) \land \text{Friend}(z) \land \text{Gives}(x, y, z) \rightarrow \text{Generous}(x)$

$\text{Owns}(\text{Eeyore}, \text{J})$ and $\text{Hunny}(\text{J})$

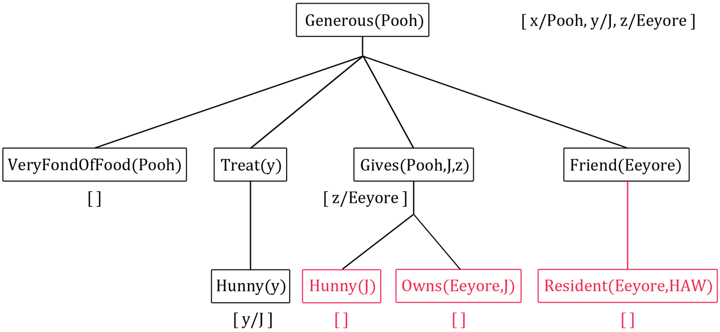
$\text{Hunny}(x) \land \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

$\text{Hunny}(x) \rightarrow \text{Treat}(x)$

$\text{Resident}(x, \text{HundredAcreWood}) \rightarrow \text{Friend}(x)$

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

$\text{VeryFondOfFood}(\text{Pooh})$



# Properties of backward chaining

Depth-first recursive proof search: space is linear in size of proof.

Incomplete due to infinite loops.

– partial fix by checking current goal against every goal on stack

Inefficient due to repeated subgoals (both success and failure).

– fix using caching of previous results (extra space)

Widely used in logic programming languages.

# Logic programming

"What's past is Prolog."

*The Tempest*, Act II, scene i

# Resolution

A method for telling whether a propositional formula is satisfiable and for proving that a first-order formula is unsatisfiable.

Yields a complete inference algorithm.

If a set of clauses if unsatisfiable, then the resolution closure of those clauses contains the empty clause (propositional logic).

# Ground Binary Resolution

$$\frac{C \lor P \qquad D \lor \neg P}{C \lor D}$$

Soundness

$C \lor P$ iff $\neg C \to P$
$D \lor \neg P$ iff $P \to D$

Therefore, $\neg C \to D$,
which is equivalent to $C \lor D$.

Note: if both $C$ and $D$ are empty, then resolution deduces the empty clause, i.e. `false`.

# Non-Ground Binary Resolution

$$\frac{C \vee P \qquad D \vee \neg P'}{(C \vee D)\ \theta}$$

where $\theta$ is the mgu of $P$ and $P'$.

The two clauses are assumed to be standardized apart so that they share no variables.

## Soundness

Apply $\theta$ to premises, then appeal to ground binary resolution.

$$\frac{C\theta \vee P\theta \qquad D\theta \vee \neg P\theta}{C\theta \vee D\theta}$$

# Example

$$\frac{\neg \text{HasHunny}(x) \lor \text{Happy}(x) \qquad \text{HasHunny}(\text{Pooh})}{\text{Happy}(\text{Pooh})}$$

with $\theta = \{x/\text{Pooh}\}$

# Factoring

$$\frac{C \lor P_1 \lor \ldots \lor P_m}{(C \lor P_i)\ \theta}$$

where $\theta$ is the mgu of the $P_i$.

Soundness

– by universal instantiation and deletion of duplicates.

# Full Resolution

$$\frac{C \vee P_1 \vee \ldots \vee P_m \qquad D \vee \neg P'_1 \vee \ldots \vee \neg P'_n}{(C \vee D)\ \theta}$$

where $\theta$ is the mgu of all $P_i$ and $P'_j$.

Soundness

– by combination of factoring and binary resolution.

To prove $\alpha$, apply resolution steps to $\mathrm{CNF}(\mathrm{KB} \wedge \neg\alpha)$.

Complete for FOL, if using full resolution or
binary resolution + factoring.

# Conversion to CNF · Example

Everyone who loves all animals is loved by someone.

$\forall x.[\forall y.\text{Animal}(y) \to \text{Loves}(x, y)] \to [\exists y.\text{Loves}(y, x)]$

- Eliminate all implications and biconditionals.

  $\forall x.\neg[\forall y.\neg\text{Animal}(y) \lor \text{Loves}(x, y)] \lor [\exists y.\text{Loves}(y, x)]$

- Move $\neg$ inwards, using $\neg\forall x.\varphi \equiv \exists x.\neg\varphi$, $\neg\exists x.\varphi \equiv \forall x.\neg\varphi$.

  $\forall x.[\exists y.\neg(\neg\text{Animal}(y) \lor \text{Loves}(x, y))] \lor [\exists y.\text{Loves}(y, x)]$

  $\forall x.[\exists y.\neg\neg\text{Animal}(y) \land \neg\text{Loves}(x, y)] \lor [\exists y.\text{Loves}(y, x)]$

  $\forall x.[\exists y.\text{Animal}(y) \land \neg\text{Loves}(x, y)] \lor [\exists y.\text{Loves}(y, x)]$

# Conversion to CNF · Example

- Standardize variables: each quantifier should use a different one.

  $\forall x.[\exists y.\text{Animal}(y) \wedge \neg\text{Loves}(x, y)] \vee [\exists z.\text{Loves}(z, x)]$

- Skolemize: a more general form of existential instantiation. Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables.[1)]

  $\forall x.[\text{Animal}(F(x)) \wedge \neg\text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x)$

[1)] No enclosing universal quantifier? Just replace with Skolem constant.

# Conversion to CNF · Example

- Drop universal quantifiers.

  $[\text{Animal}(F(x)) \wedge \neg\text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x)$

- Distribute $\vee$ over $\wedge$.

  $[\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)] \wedge [\neg\text{Loves}(x, F(x)) \vee \text{Loves}(G(x), x)]$

# Resolution algorithm

```
function PL-RESOLUTION(KB, α) returns true or false
   inputs: KB, the knowledge base, a sentence in propositional logic
           α, the query, a sentence in propositional logic

   clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
   new ← { }
   loop do
      for each pair of clauses C_i, C_j in clauses do
         resolvents ← PL-RESOLVE(C_i, C_j)
         if resolvents contains the empty clause then return true
         new ← new ∪ resolvents
      if new ⊆ clauses then return false
      clauses ← clauses ∪ new
```

**returns the set of all possible clauses obtained by resolving its two inputs** (annotation pointing to PL-RESOLVE$(C_i, C_j)$)

# Example · Winnie-the-Pooh CNF

$\neg \text{VeryFondOfFood}(x) \vee \neg \text{Treat}(y) \vee \neg \text{Friend}(z) \vee \neg \text{Gives}(x, y, z) \vee \text{Generous}(x)$

$\text{Hunny}(\text{J})$

$\text{Owns}(\text{Eeyore}, \text{J})$

$\neg \text{Hunny}(x) \vee \neg \text{Owns}(\text{Eeyore}, x) \vee \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

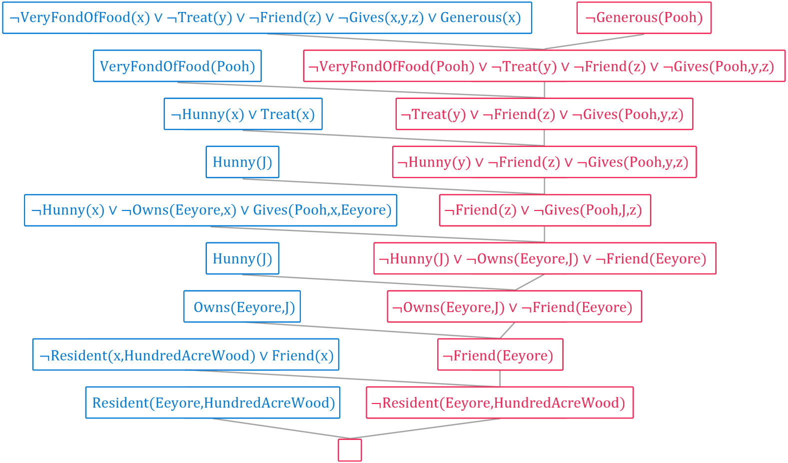
$\neg \text{Hunny}(x) \vee \text{Treat}(x)$

$\neg \text{Resident}(x, \text{HundredAcreWood}) \vee \text{Friend}(x)$

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

$\text{VeryFondOfFood}(\text{Pooh})$

# Example · Resolution proof



# Summary

- Forward chaining
- Backward chaining
- Resolution