Logical Representation & Reasoning: Predicate Logic

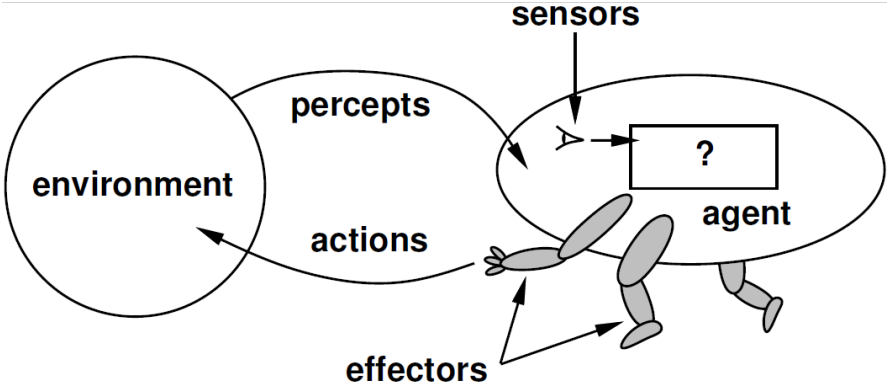
Part I: Motivation & Syntax

Course on AI@IID

Slides adapted from Mannes Poel by M. Birna van Riemsdijk

Designing rational agents

- Knowledge/logic based approach to designing and implementing the ? component of rational agents.



- Needed: logical modelling of the environment, the knowledge base *KB* of the agent and inference (reasoning) with this knowledge

Looking back Propositional Logic

- Not very expressive.
- Does not fit the way humans think and reason and we are the designers of intelligent agents.
- Knowledge such as “Every adult is older than 17” is hard to translate in Propositional Logic. Must be done by enumeration:
  - $adult\_peter \Rightarrow age\_peter\_larger\_than\_17$
  - $adult\_mary \Rightarrow age\_mary\_larger\_than\_17$
  - …….

(First-Order) Predicate Logic

- Knowledge is represented by predicates, variables, quantifiers, and logical connectives:
- Examples:
  - $\forall x\ Adult(x) \Rightarrow Age(x) > 17$
  - $\forall x\ Human(x) \Rightarrow \exists y\ Human(y) \wedge Mother(y, x)$
  - Different modeling in which *Mother* is a function
  - $\forall x\ Human(x) \Rightarrow \exists y\ Human(y) \wedge Mother(x) = y$

Syntax (1)

- **Constants:** *John, Mary, 2, UT, AI*
- **Predicates:** *Parent, Daughter, Neighbor, > , Student, Takes, Smart*
- **Functions:** *Sqrt, Mother, Father, Teacher*
- **Variables:** *x,y,z*
- **Connectives:** $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$
- **Equality:** *=*
- **Quantifiers:**
  - $\forall$*:* for all, universal quantifier
  - $\exists$: exists, existential quantifier

Syntax (2)

- **Atomic Sentence:**
  - *predicate($term_1$,…,$term_n$)*
  - *$term_1$=$term_2$*
- **Term:**
  - *function($term_1$,…,$term_k$)*
  - *constant*
  - *variable*
- Examples:

  Location(Wumpus, Square(1,3))

  North(Wumpus, Agent)

Syntax (3)

- **Complex Sentences:**
  - $\neg S, S_1 \wedge S_2, S_1 \vee S_2, S_1 \Rightarrow S_2, S_1 \Leftrightarrow S_2$
- **Quantifiers:**
  - $\forall x\ S(x)$
  - $\exists y\ S(y)$

- Example:

$$\forall x\ Passes(x, AI) \Rightarrow Smart(x)$$
$$\forall x\ Human(x) \Rightarrow \exists y\ Loves(x, y)$$

Exercise

- How do we model the sentence “All students who take the course on AI are smart”?

- $\forall x\ Student(x) \wedge Takes(x, AI) \Rightarrow Smart(x)$

Exercise

- How do we model the sentence “There is a student who takes AI and is smart.”

- ~~$\exists x\ Student(x) \wedge Takes(x, AI) \Rightarrow Smart(x)$~~

- $\exists x\ Student(x) \wedge Takes(x, AI) \wedge Smart(x)$

Common mistakes to avoid

- Typically, $\wedge$ is the main connective with $\exists$
- Common mistake: using $\Rightarrow$ as the main connective with $\exists$.

Logical Representation & Reasoning: Predicate Logic

Part II: Semantics

Course on AI@IID

Slides adapted from Mannes Poel by M. Birna van Riemsdijk

Semantics

- Just like for propositional logic, a model in predicate logic must provide the information required to determine if any syntactically correct sentence is *true* or *false.*
- For predicate logic a model is however much more complex than for propositional logic.
  - Atomic sentences have more structure: a true/false assignment for atoms is not enough
  - We need to find a way to interpret quantifiers over variables
- What does a model in predicate logic look like?

Models

- Model contains objects (domain elements) and relations among them
- Interpretation maps syntax to elements in the semantic domain, specifying referents for

constant symbols	→	objects
predicate symbols	→	relations
function symbols	→	functional relations

- An atomic sentence *predicate*($term_1$,...,$term_n$) is true iff the objects referred to by $term_1$,...,$term_n$ are in the relation referred to by *predicate*

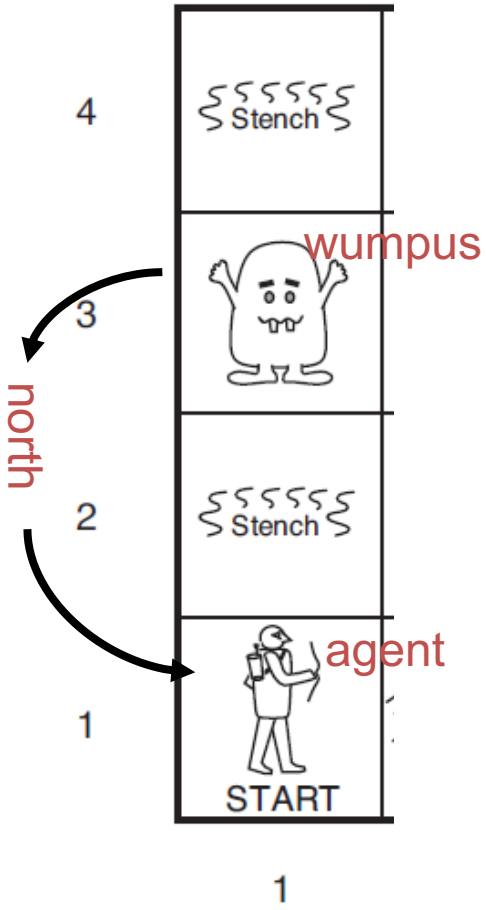
Example

Syntax

- North(Wumpus, Agent)

- An interpetation defines the mapping of syntax elements to elements in the domain of interpretation
- Binary predicate "North" is mapped to the binary relation "north", and the constants Wumpus and Agent are mapped to the individuals wumpus and agent.
- We have: $(wumpus, agent) \in north$

Semantics



Exercise

- Determine the models for the sentences:
  $\forall x\ Student(x) \wedge Takes(x, AI) \Rightarrow Smart(x)$
  $\forall x\ Passes(x, AI) \Rightarrow Smart(x)$
  $\forall x\ Human(x) \Rightarrow \exists y\ Loves(x, y)$

- Well that is much harder than for Prop. Logic.

- Can be reduced to propositional logic if we can enumerate all possibilities.

Semantics of Quantifiers

- $\forall x\ P(x)$ is *true* in model *m* iff *P* is *true* in model *m* for every instantiation of x. So it is equivalent to

$$P(A) \wedge P(B) \wedge \cdots \wedge P(K)$$

  where *A,B,..,K* are the possible instantiations of x.

- $\exists x\ P(x)$ is *true* in model *m* iff *P* is *true* in *m* for at least one instantiation of x.

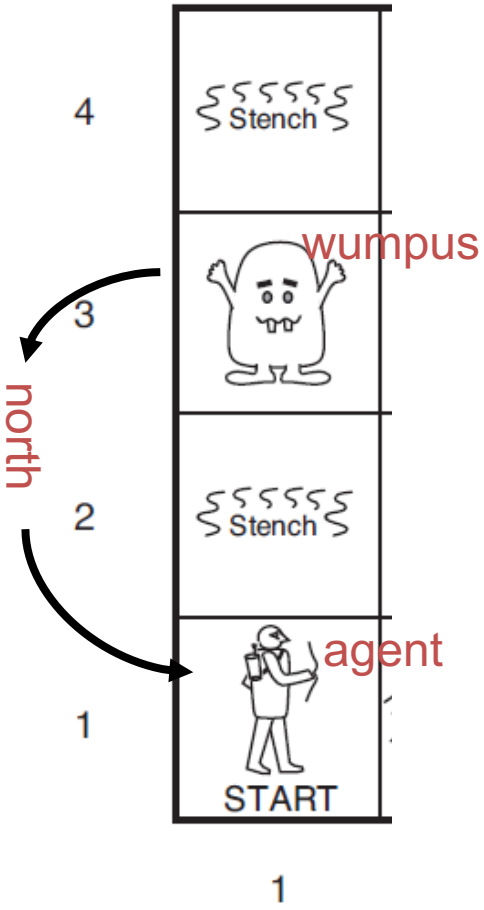
Example

Syntax

- $\exists x$ North(x, Agent)

- An interpetation defines the mapping of syntax elements to elements in the domain of interpretation
- Map/instantiate the variable x to the individual wumpus.
- We have: $(wumpus, agent) \in north$
- Thus the sentence holds in this model.

Semantics



Logical Representation & Reasoning: Predicate Logic

Part III: Unification

Course on AI@IID

Slides adapted from Mannes Poel by M. Birna van Riemsdijk

Unification

- In applying the resolution rule for predicate logic, you have to determine the equality of two literals
- Say, *Knows*(*John*, x) and *Knows*(*John*, *Jane*)
- They are not the same but they can be made the same if we substitute *Jane* for x
  - Unifier: a substitution that makes two literals equal
  - Notation: $\{x/Jane\}$
- This is called *unification*
- x is (implicitly) universally quantified and thus applies to every object in the domain, so also to Jane.
- Unification is a fundamental computational mechanism in Prolog!

Unification

1. If T_1 and T_2 are constants, then
   T_1 and T_2 unify if they are the same constant
2. If T_1 is a variable and T_2 is any type of term, then T_1 and T_2 unify, and T_1 is instantiated to T_2. (and vice versa)
3. If T_1 and T_2 are complex terms then they unify if:
   a) They have the same functor and arity (= number of arguments), and
   b) all their corresponding arguments unify, and
   c) the variable instantiations are compatible.

Unification: examples

- *Knows*(*John*, *x*) and *Knows*(*y*, *Jane*): {*x*/*Jane*, *y*/*John*}
  - Unified literal thus *Knows*(*John*, *Jane*)

Take care with functions

- *Loves*(*x*, *S*(*x*)) and *Loves*(*John*, *z*): OK
  - Start with *x*/*John* and *z*/*S*(*x*), then (because you already said *x*/*John*): *z*/*S*(*John*)
  - Substitution thus {*x*/*John*, *z*/*S*(*John*)}
  - Unified literal thus *Loves*(*John*, *S*(*John*))

Most General Unifier (mgu)

Most general unifier: a unifier for which it holds that every other unifier is an "extension".

- *Knows*(*John*, *x*) and *Knows*(*y*, *z*):
  - A unifier: {*x/Mary, y/John, z/Mary*} yields the literal *Knows(John, Mary)*
  - But this is not the most general unifier! We did not have to commit to substitute x and z with Mary.
  - mgu: {*x*/*z*, *y*/*John*}, with unified literal thus *Knows*(*John*, *z*)

Unification: two more examples involving functions

- $Loves(x, S(x))$ and $Loves(y, z)$: OK, $\{y/x, z/S(x)\}$
  - Unified literal thus $Loves(x, S(x))$
- $Loves(x, S(x))$ and $Loves(y, y)$: fails because in general $x \neq S(x)$
- The latter case can be excluded by means of the *occurs check*: check whether a variable occurs as argument of a function (no matter how deeply nested) *with which it has to be unified*.
  - Only when occurs check okay then unification possible

Logical Representation & Reasoning: Predicate Logic

Part IV: Reasoning

Course on AI@IID

Slides adapted from Mannes Poel by M. Birna van Riemsdijk

Inference

- Theorem proving: applying rules to derive a proof of KB $\models \alpha$ without model checking.

- Well known rule is **Universal Instantiation**:

$$\frac{\forall x\ P(x)}{Subst(\{x/A\}, P(x))}$$

Substituting A for x (denoted by $\{x/A\}$) in P, equals $P(A)$

Inference

Well known rule is **Existential Instantiation**:

$$\frac{\exists x\ P(x)}{Subst(\{x/A\}, P(x))}$$

Substituting A for x in P in which A is a constant (unique new identifier) that does not appear elsewhere in the knowledge base.

Concepts

- **Logical equivalence:** $\alpha \equiv \beta$ if $M(\alpha) = M(\beta)$**.**
- Examples: $p \ \wedge q \equiv q \wedge p$.
- **Validity:** A sentence α is valid if it is true in all models (a.k.a. **tautologies**)
- **Satisfiability:** a sentence α is satisfiable if it is true in some model, i.e. $M(\alpha) \neq \emptyset$
- Example: $KB \models \alpha$ if $KB \wedge \neg\alpha$ is unsatisfiable, i.e. $M(KB \wedge \neg\alpha) = \emptyset$. (Proof by contradiction.)

Equivalence

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge$$
$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee$$
$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge$$
$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee$$
$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \quad \text{implication elimination}$$
$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination}$$
$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \quad \text{de Morgan}$$
$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \quad \text{de Morgan}$$
$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee$$
$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge$$

Equivalence for quantifiers

- $\forall x \neg P \equiv \neg \exists x \; P$
- $\neg \forall x \; P \equiv \exists x \neg P$
- $\forall x \; P \equiv \neg \exists x \neg P$
- $\exists x \; P \equiv \neg \forall x \neg P$

Logical Inference for Predicate Logic

1. Transforming to CNF
2. Unification
3. Resolution

Resolution rule

$$\frac{l_1 \vee l_2 \ldots \vee l_k \qquad m_1 \vee m_2 \ldots \vee m_n}{Subst(\theta, l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \ldots \vee m_n)}$$

Where $Unify(l_i, \neg m_j) = \theta$.

One technical aspect: remove multiple copies of the same literal.

Transforming to CNF

The recipe becomes more complicated because we have to get rid of the quantifiers

1. Replace all $A \Leftrightarrow B$ by $(A \Rightarrow B) \wedge (B \Rightarrow A)$
2. Replace all $A \Rightarrow B$ by $\neg A \vee B$
3. Move negation signs inwards starting from the outside until every negation sign is directly in front of a literal
4. Remove quantifiers
5. Distribute $\vee$ over $\wedge$ until you obtain CNF

Move negation inside

Remember:

- $\neg \forall x\ P \equiv \exists x\ \neg P$
- $\neg \exists x\ P \equiv \forall x\ \neg P$

Remove quantifiers

- Existentially quantified variables are replaced by constants or functions: "Skolemisation"
- Then every remaining variable is universally quantified
- Thus, universal quantifiers can be dropped, i.e., formulas become implicitly universally quantified
  - Also for rules in Prolog!

Simple Skolemisation

- General idea: if $\exists x\ P(x)$ is true, $P(S)$ is true
  - S is a constant (called the Skolem constant)
  - S may not appear elsewhere in the proof
  - Example: There is a man -> Man(John)
- We call this *simple Skolemisation*
- Simple Skolemisation works as long as the existentially quantified variable x is not within the scope of a universal quantifier.

What is the problem

- $\forall x\ Human(x) \Rightarrow \exists y\ Loves(x, y)$

Would be replaced by

$$\forall x\ Human(x) \Rightarrow Loves(x, S)$$

Is this correct?

- Problem y depends on x, is a function of x.
- Solution introduce Skolem function:

$$\forall x\ Human(x) \Rightarrow Loves(x, S(x))$$

Skolemisation

- The existentially quantified variable x is within the scope of a universal quantifier $\forall y$ if:
  - $\exists x$ follows after $\forall y$, and
  - there is at least one predicate of which both x and y are arguments
- When simple Skolemisation is incorrect, use a Skolem function instead of a Skolem constant
- If $\forall y\ \exists x\ P(x, y)$ is true, $\forall y\ P(S(y), y)$ is true
  - The Skolem function S may not appear elsewhere in the *KB*

Resolution rule

$$\frac{l_1 \vee l_2 \ldots \vee l_k \qquad m_1 \vee m_2 \ldots \vee m_n}{Subst(\theta, l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \ldots \vee m_n)}$$

Where $Unify(l_i, \neg m_j) = \theta$:

$l_i = Grandparent(G, John)$
$m_j = \neg Grandparent(x, y)$

$\theta = \{x/G, y/John\}$

One technical aspect: remove multiple copies of the same literal.

Proof by Resolution for Predicate Logic

- Goal is to prove: $KB \models \alpha$

1. Add $\neg\alpha$ to KB and prove $KB \wedge \neg\alpha \models \mathit{false}$
2. Write $KB \wedge \neg\alpha$ in CNF
3. Apply *resolution* rule until no **new** clause can be added anymore or *false* is derived for instance due to $p \wedge \neg p$ (*empty clause*).

Proof by resolution Predicate Logic

- It is sound: if it is proven by resolution that α follows from *KB* then $KB \models \alpha$
- It is **refutation-complete**: If $KB \models \alpha$ then it can be proven by resolution that α follows from *KB.* Resolution cannot be used to generate all logical consequences of a set of sentences.

What did we learn

- Basic logical formalisms for modeling knowledge:
  - Propositional Logic: $p, q, p \vee q, p \wedge q, \neg p, p \Rightarrow q,$
  - Predicate Logic: $P(x, y), \forall x\ Q(x), \forall x \exists y\ P(x, y)$
- Basic formalism for computational logic inference:
  - Resolution, which is sound and (refutation-)complete.

Logical Representation & Reasoning: Predicate Logic

Part V: Example Inference

Course on AI@IID

Slides adapted from Mannes Poel by M. Birna van Riemsdijk

Resolution in first-order logic Example

  - Your grandparent is the parent of your parent
  - John has a grandparent
- From this, prove by resolution:
  - John has a parent
- Formalize premises:

$\forall x, y\ Grandparent(x, y) \Rightarrow\ \exists z\ Parent(x, z) \wedge Parent(z, y)$
$\exists x\ Grandparent(x, John)$

- Formalize conclusion and put negation in front:
$\neg\ \exists x\ Parent(x, John)$

Resolution in first-order logic, example

- Be careful with variables: the *x* in one sentence and the *x* in another sentence are not the same variable!
- This becomes relevant when you start instantiating variables
- Precaution: standardise apart
  - Or: rename variables so that they are different per sentence
- Rename variable in second premises:
  $\forall x, y\ Grandparent(x, y) \Rightarrow \exists z\ Parent(x, z) \wedge Parent(z, y)$
  $\exists v\ Grandparent(v, John)$
- Rename variable in negated conclusion:
  $\neg\exists w\ Parent(w, John)$

Resolution in first-order logic, example

- First remove $\Rightarrow$ (implication elimination):

$$\forall x, y\ Grandparent(x, y) \Rightarrow \exists z\ Parent(x, z) \wedge Parent(z, y)$$

$$\downarrow$$

$$\forall x, y\ \neg Grandparent(x, y) \vee [\exists z\ Parent(x, z) \wedge Parent(z, y)]$$

- Note the position of the negation sign!
- We are now in a position to remove the quantifiers
  - There are no negation signs to be pushed inwards in this example

Resolution in first-order logic, example

First work away $\exists$:

- $\exists v\ Grandparent(v, John)$

  Simple Skolemisation works: (G is a new constant)

  $$Grandparent(G, John)$$

- $\neg\exists w\ Parent(w, John)$

  Negation before quantifier, hence quantifier swap:

  $$\forall w\ \neg Parent(w, John)$$

Resolution in first-order logic, example

(Still working away ∃)

- $\forall x, y \ \neg Grandparent(x, y) \vee$
  $[\ \exists z\ Parent(x, z) \wedge Parent(z, y)\ ]$
  - z within the scope of both x and y
  - Simple Skolemisation is not allowed
  - Hence: introduce a Skolem function with two (!) arguments:
- $\forall x, y \ \neg Grandparent(x, y) \vee$
  $(Parent(x, S(x, y)) \wedge Parent(S(x, y), y))$

Resolution in first-order logic, example

- Now only universal quantifiers remain; drop them

  $Grandparent(G, John)$
  $\neg Parent(w, John)$
  $\neg Grandparent(x, y) \vee (Parent(x, S(x, y)) \wedge Parent(S(x, y), y))$
- First two in CNF, third isn't

Resolution in first-order logic, example

$$\neg Grandparent(x, y) \vee$$
$$[Parent(x, S(x, y)) \wedge Parent(S(x, y), y)]$$

(distributivity)

$$\left[\neg Grandparent(x, y) \vee Parent\left(x, S(x, y)\right)\right] \wedge$$
$$[\neg Grandparent(x, y) \vee Parent(S(x, y), y)]$$

CNF!

Resolution in first-order logic, example

- We have, conjuncts taken apart:

$$Grandparent(G, John)$$
$$\neg Parent(w, John)$$
$$\neg Grandparent(x, y) \vee Parent(x, S(x, y))$$
$$\neg Grandparent(x, y) \vee Parent(S(x, y), y)$$

- Last steps, resolution, in “animated” form

Resolution in first-order logic, example

$Grandparent(G, John)$

$\neg Parent(w, John)$

$\neg Grandparent(x, y) \vee Parent(x, S(x, y))$

$\neg Grandparent(x, y) \vee Parent(S(x, y), y)$

unify {x/*G*, y/*John*} and apply resolution

$x = G$
$y = John$

Resolution in first-order logic, example

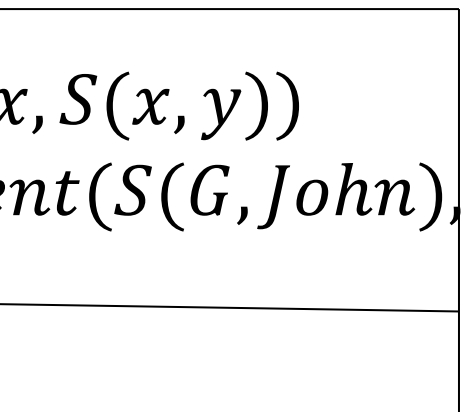
$Grandparent(G, John)$

$\neg Parent(w, John)$

$\neg Grandparent(x, y) \vee Parent(x, S(x, y))$

$\neg Grandparent(G, John) \vee Parent(S(G, John), John)$

$Parent(S(G, John), John)$



unify $\{w/S(G, John)\}$

$x = G$
$y = John$
$w = S(G, John)$

Resolution in first-order logic, example: done!

$Grandparent(G, John)$

$\neg Parent(S(G, John), John)$

$\neg Grandparent(x, y) \vee Parent(x, S(x, y))$

$\neg Grandparent(G, John) \vee Parent(S(G, John), John)$

$Parent(S(G, John), John)$

contradiction!

$x = G$
$y = John$
$w = S(G, John)$