



Linear Resolution and Introduction to First Order Logic

Michael Leuschel

Softwaretechnik und Programmiersprachen



Recap: CNF & Resolution

- Conjunctive Normal Form: set of **clauses**
 - clause = disjunction of **literals**
 - literal = p or $\neg p$
 - Example: $\{ \neg p \vee q, \text{ wet} \vee \neg \text{rains} \vee \neg \text{outside} \}$
- Resolution
 - From $p \vee \alpha$ and $\neg p \vee \beta$ derive a new logical consequence $\alpha \vee \beta$
- Proof by refutation for $S \Rightarrow \{L\}$:
 - Show that $S \cup \{\neg L\}$ is inconsistent by obtaining the contradiction \square by **resolution**



Linear Resolution

T:
 $p \vee \neg q$
q

D: $\neg p$

D': $\neg q$

- 1. Take a theory **T** in CNF with Horn clauses, but no denials
- 2. Take a denial **D**
- 3. Resolve the denial with one clause from T
 - you obtain a new denial **D'**
- 4. Set $D := D'$ (forget old denial) and restart at step 2 until you reach a contradiction
- Observations:
 - Negative literals: always from D, Positive literals always from T, T remains unchanged

Linear Resolution and Prolog

umbrella \leftarrow rain \wedge outside

```
umbrella :-  
    rain, outside.  
rain.  
outside.
```

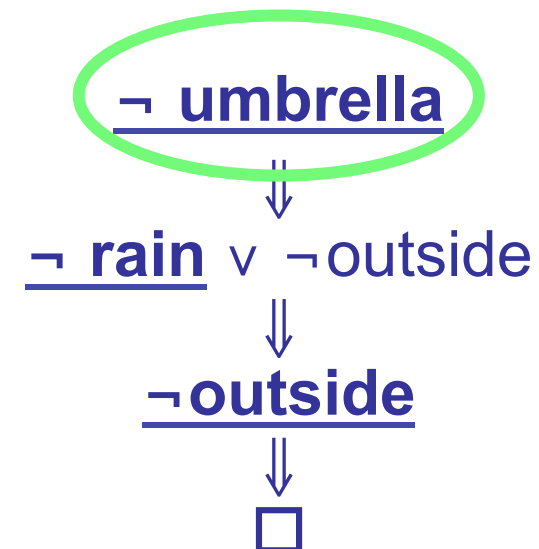
```
?- umbrella.
```

```
yes
```

umbrella \vee \neg rain \vee
 \neg outside

rain
outside

*Linear =
One can forget the past*





Non Linear Resolution

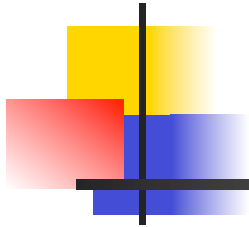
- One day Tokusan told his student Gantō,
*“I have two monks who have been here for many years.
Go and examine them.”*
- Gantō picked up an ax, saying,
*“If you say a word I will cut off your heads;
and if you do not say a word, I will also cut off your
heads.”*

- 1) cut $v \rightarrow \text{say}$
- 2) cut $v \text{ say}$

Non-linear resolution required
to show that “cut” is a logical
consequence!

Observe: 2) not a Horn clause

*Linear =
One can forget the past*



First-Order Logic

19/04/2011



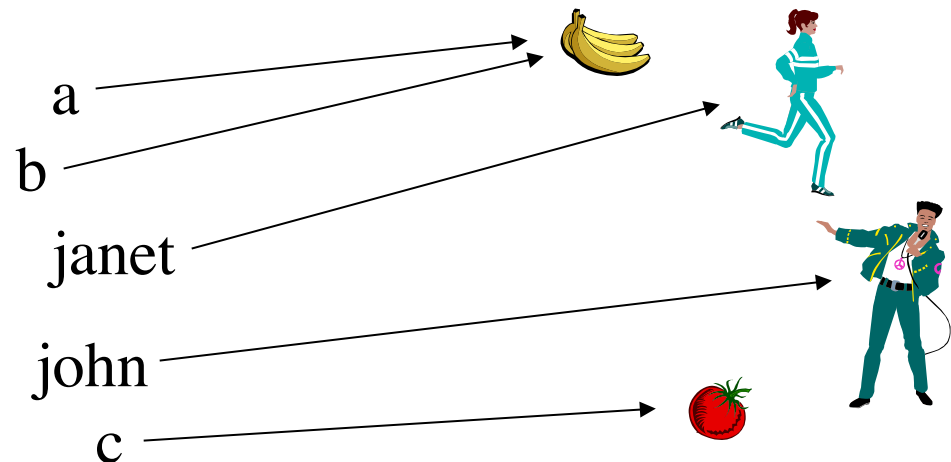
Why a more expressive logic?

- Example:
 - John loves all girls
 - Janet is a girl
 - *Therefore*, John loves Janet
- Propositional Logic:
 - $\{j_loves_all_girls, janet_is_girl\} \not\Rightarrow \{j_loves_janet\}$
 - But: argument above still valid
- \rightarrow We have to be able to talk about **objects/individuals**

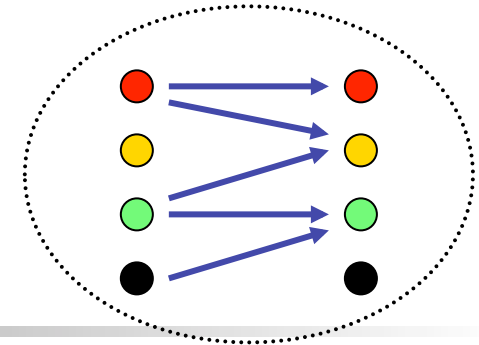
Addition 1: Constants

- **Constants:** a,b,c, john, janet
 - Start with lowercase letter
 - Do not stand for a truth-value (true/false)
 - Represent a particular object from the “real world”

(mathematically: interpretations will map each constant to an element from a domain)



Addition 2: Predicates

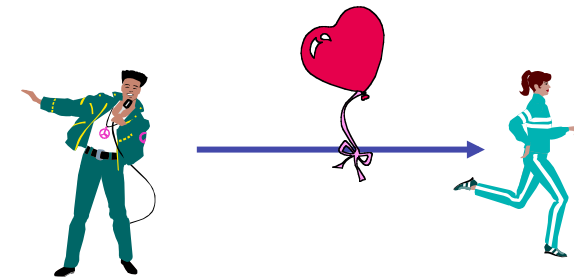


binary
relation

- **Predicates:** girl(), loves(,), ...
- Start with lowercase letter
- Represent **relations** between objects
- Have an arity
 - loves(,): 2
 - girl(): 1
 - rains: 0

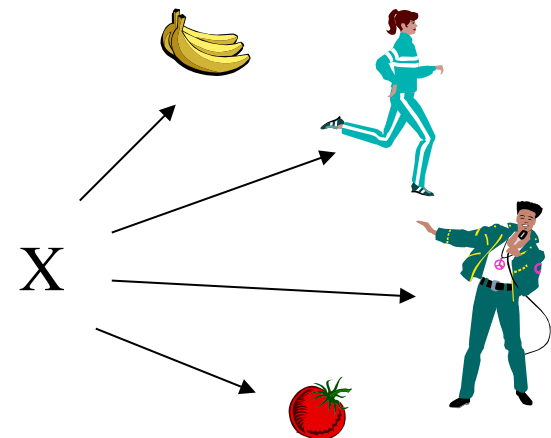
(propositions are just a special case)

loves(john,janet)



Addition 3: Variables

- **Variables: X, Y, Z, \dots**
 - Start with uppercase letter
 - Do not have a truth-value either
 - Can stand for “any” object from the “real world”





Addition 4: Quantifiers

- **Universal** quantifier \forall (for all)
 - Example: $\forall X \text{ loves}(X,X)$
 - Intuitively: “Everybody loves himself”
 - $(\forall X F)$: means that for **all possible values** of X the formula F has to be true
- **Existential** quantifier \exists (there exists)
 - Example: $\exists X \text{ girl}(X)$
 - Intuitively: “There exists a girl”
 - $(\exists X F)$: means that we can find **one value** (at least) for X such that the formula F is true



Exercise: Dylan

- Translate into logic:
 - *You can fool **some** of the people **all** of the time.*
 - *You can fool **all** of the people **some** of the time.*
 - *But you **cannot** fool **all** of the people **all** of the time.*

- Use predicate:
 - fool(X, T): you can fool X at time T



Exercise: Solution

- Translate into logic:
 - You can fool **some** of the people **all** of the time.
 - You can fool **all** of the people **some** of the time.
 - But you **cannot** fool **all** of the people **all** of the time.

- A Solution:
 - $\exists X \forall T \text{ fool}(X, T)$
 - $\forall X \exists T \text{ fool}(X, T)$ (also acceptable: $\exists T \forall X \text{ fool}(X, T)$)
 - $\neg (\forall X \forall T \text{ fool}(X, T))$

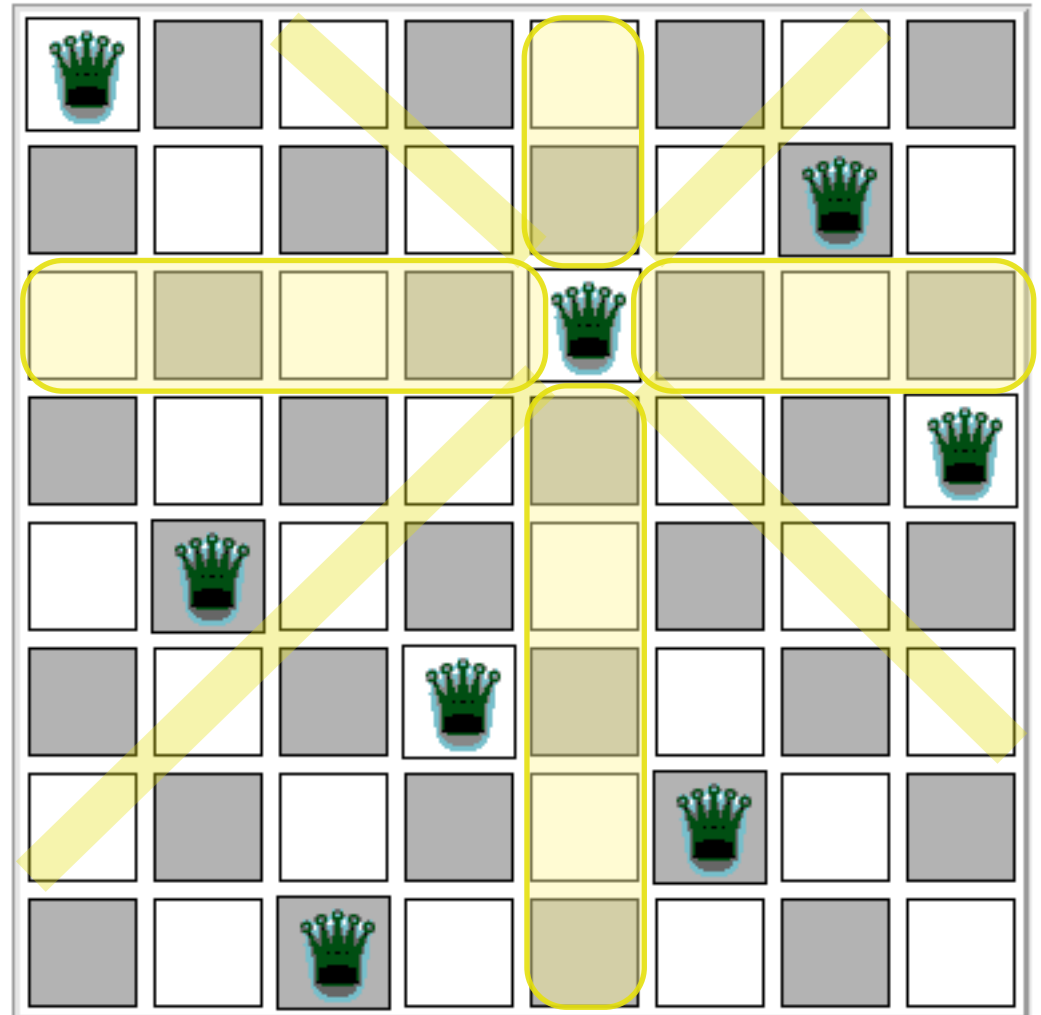


Exercise 2

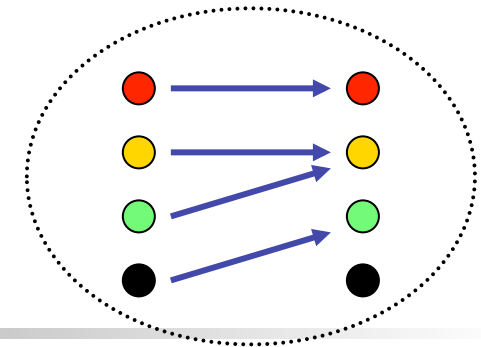
- Graph Colouring
 - Predicates:
 - `colour(node,col)`
 - `link(node1,node2)`
 - `=/2`
 - adjacent nodes have different colour

Exercise 3

- N-Queens
 - Predicates:
 - `queen(i,j)`



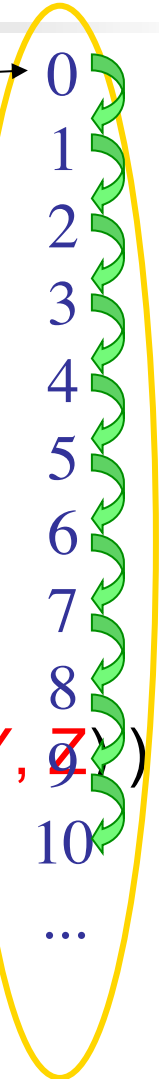
Addition 5: Function Symbols



unary
function

- **Function symbols:**
 - succ(), **cons**(,), ...
 - Start with lowercase letter
 - Represent total **functions** between objects
 - Have an arity
 - **cons**(,): 2
 - succ(): 1
 - john: 0 (constants are just a special case)

Natural Numbers

- Constant **0** → 0
 - Function symbol **succ()** → succ ↻
 - Describing natural numbers:
 - $\text{nat}(0)$
 - $\forall X (\text{nat}(\text{succ}(X)) \leftarrow \text{nat}(X))$
 - Describing addition:
 - $\forall X \text{ plus}(0, X, X)$
 - $\forall X Y Z (\text{plus}(\text{succ}(X), Y, \text{succ}(Z)) \leftarrow \text{plus}(X, Y, Z))$
- 



Exercise

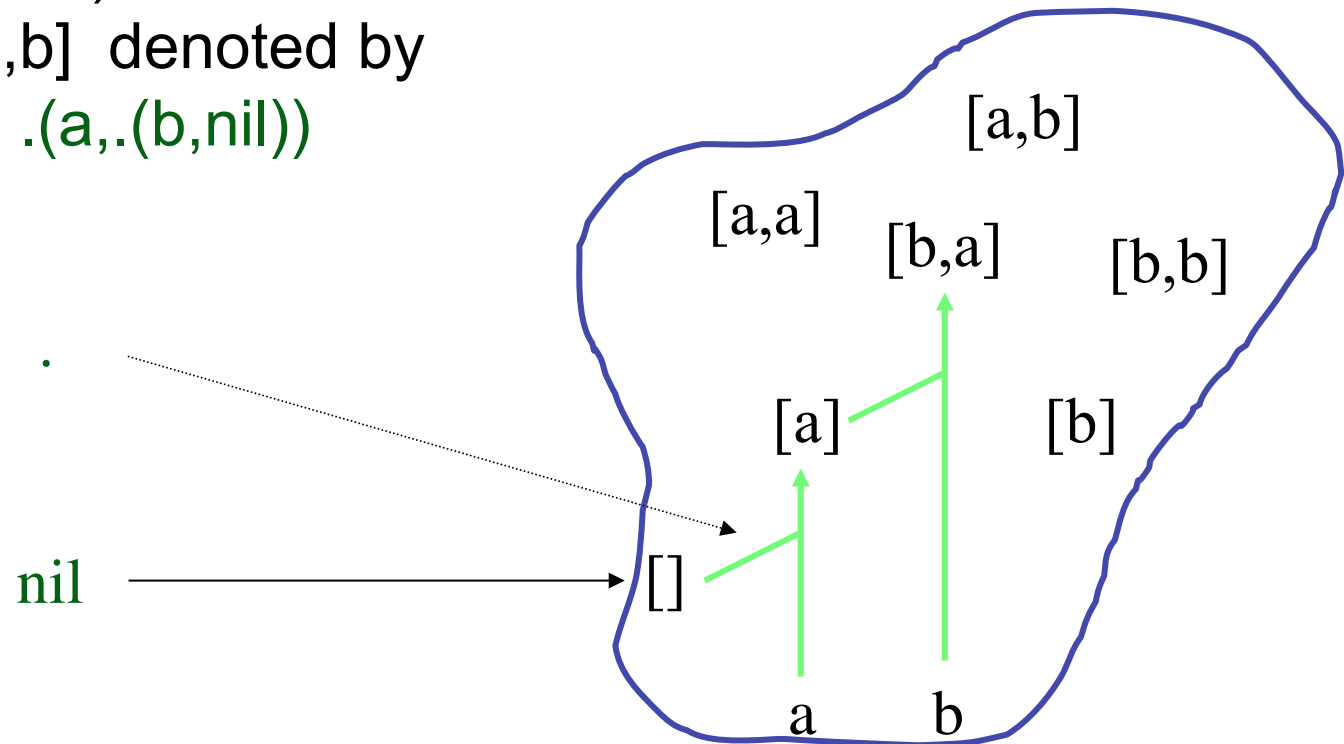
- Define (in terms of plus/3)
 - Subtraction
 - Multiplication
- Run it in Prolog



Representing Lists

- Constant **nil**: empty list
 - Function symbol **.(_,_)**
 - **.(H,T)**: list with first element H and tail T
- [a,b] denoted by
 $.(a,.(b,nil))$

$\forall X \text{ app}([], X, X)$



Summary

1. First-Order Logic

- constants, functions, variables
(**tom** **mother**(**tom**) **X**)
- relations
(**human**(.) **married**(. , .))
- quantifiers
(**∀** **∃**)

human(sokrates)

∀ X. human(X) **→ mortal**(X)

variables over objects

0. Propositional Logic

- propositions
(**basic units**: either true or false)
- logical connectives
(**∨** **∧** **¬** **→**)

rains

rains **→ carry_umbrella**

rains **∨ ¬ rains** **p** **∨ ¬ p**

no variables



What definitely to know for the exam

- Linear Resolution
- Ingredients of First Order Logic (FOL)
 - Constants, predicates, quantifiers, function symbols
- Translating natural language into FOL
 - Tutorial
- How to represent data-structures in FOL
 - Natural numbers, lists
 - Trees (later in the course)