Chapters 2&3: A Representation and Reasoning System

- **Lecture 1** Representation and Reasoning Systems. Datalog.
- **Lecture 2** Semantics.
- **Lecture 3** Variables, queries and answers, limitations.
- **Lecture 4** Proofs. Soundness and completeness.
- **Lecture 5** SLD resolution.
- **Lecture 6** Proofs with variables. Function Symbols.

Representation and Reasoning System

A Representation and Reasoning System (RRS) is made up of:

- formal language: specifies the legal sentences
- semantics: specifies the meaning of the symbols
- reasoning theory or proof procedure: nondeterministic specification of how an answer can be produced.

Implementation of an RRS

An implementation of an RRS consists of

- **language parser:** maps sentences of the language into data structures.
- **reasoning procedure:** implementation of reasoning theory + search strategy.

Note: the semantics aren't reflected in the implementation!

From a task to an RRS

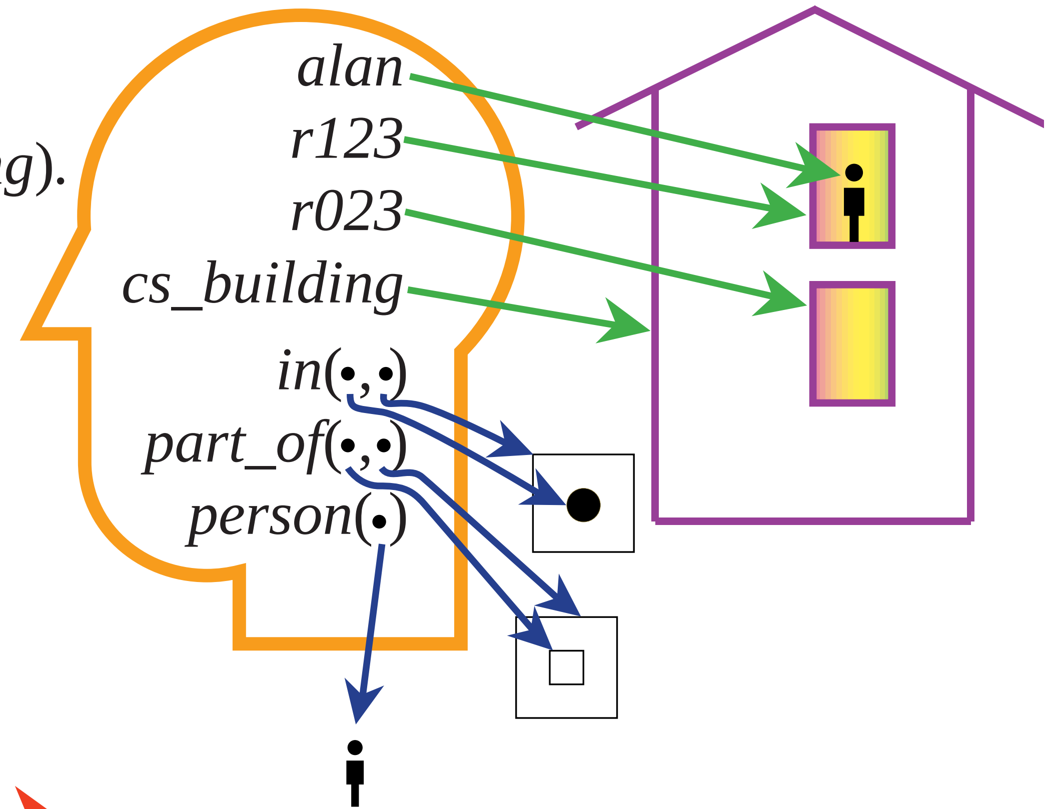
- The domain: What are the tasks to be performed w.r.t the domain?
- Analysis of the domain: Objects and relations in the domain. (Ontology and conceptualization)
- Symbolic representation of the domain: Symbols denoting objects and relations in the world.
- Knowledge representation: Describing what is true in the domain and how to solve problems in the domain.
- Testing the RRS: Asking the RRS some questions which prompt it to reason with its knowledge.

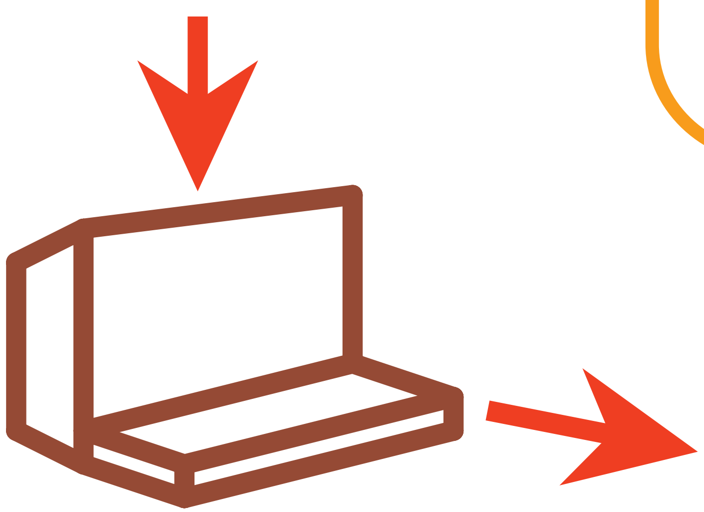
Using an RRS

1. Begin with a task domain.
2. Distinguish those things you want to talk about (the ontology).
3. Choose symbols in the computer to denote objects and relations.
4. Tell the system knowledge about the domain.
5. Ask the system questions.

Role of Semantics in an RRS

in(*alan*,*r123*).
part_of(*r123*,*cs_building*).
in(*X*,*Y*) ←
    part_of(*Z*,*Y*) ∧
    in(*X*,*Z*).





in(*alan*,*cs_building*)

Simplifying Assumptions of RRSs

- Starting with simplifications can be a successful strategy for solving complex problems!
- Three types of simplifying assumptions:
  - Assumptions w.r.t. the agent
  - Assumptions w.r.t. the environment
  - Assumptions w.r.t. relations between the agent and the environment.

Simplifying Assumptions of Initial RRS

An agent's knowledge can be usefully described in terms of *individuals* and *relations* among individuals.

An agent's knowledge base consists of *definite* and *positive* statements.

The environment is *static*.

There are only a finite number of individuals of interest in the domain. Each individual can be given a unique name.

$\Longrightarrow$ Datalog

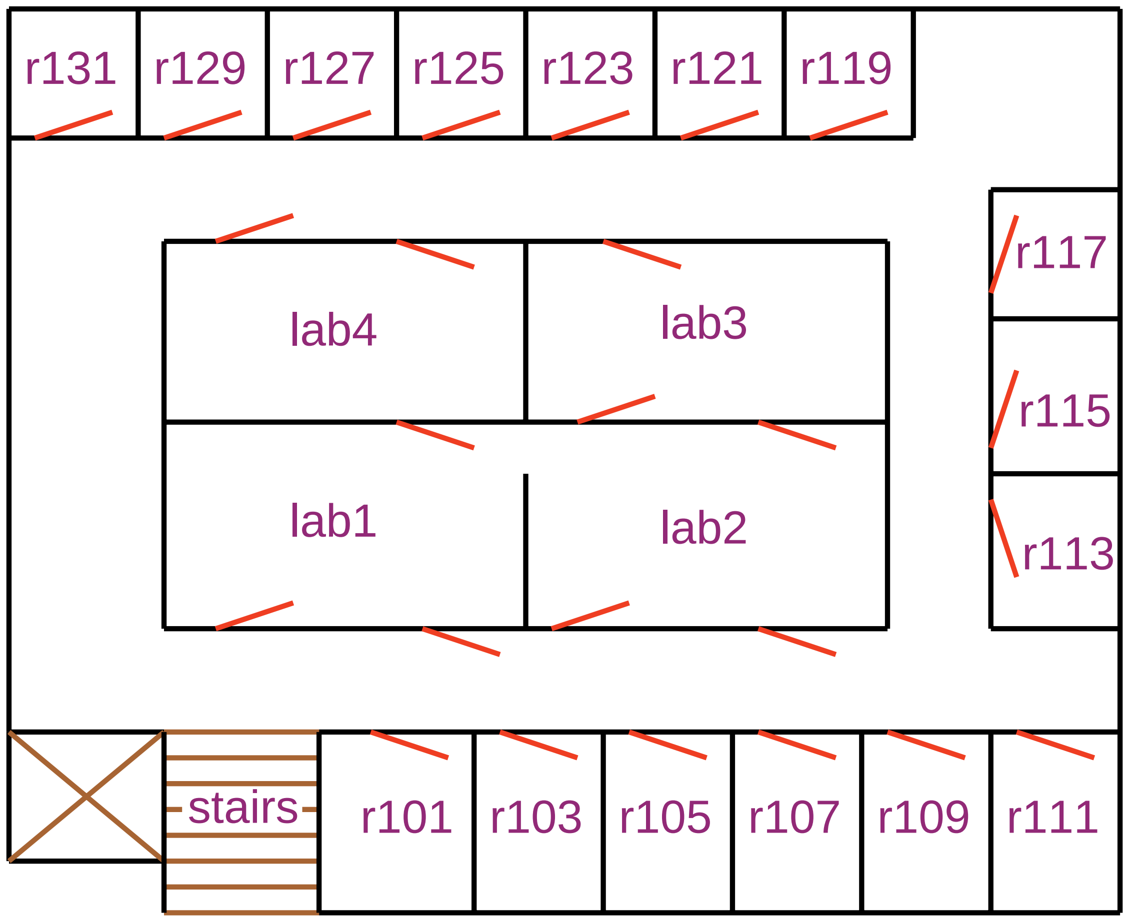
The Definite Knowledge Assumption

- Pieces of knowledge are not indeterminate, vague or negative:
  - Craig is in r113 or in lab2.
    If Craig is in the cs-building, he is in lab3.
  - Craig is near room r117.
  - Craig is not in room r117.

The Static Environment Assumption

- Change of the environment is not subject of the agent's, i.e., the RRS's, reasoning tasks.
- Slogan: *"The environment does not change."*
  - Yesterday, Craig was in lab2.
  - The room-numbering system is static.
  - The coffee machine has not been moved to another room.

Domain for Delivery Robot



Syntax of Datalog

variable starts with upper-case letter.

constant starts with lower-case letter or is a sequence of digits (numeral).

predicate symbol starts with lower-case letter.

term is either a variable or a constant.

atomic symbol (atom) is of the form p or $p(t_1, \ldots, t_n)$ where p is a predicate symbol and t_i are terms.

The Syntax of Datalog (cont)

- **body** is of the form

  $a_1 \wedge \ldots \wedge a_m$,

  where each a_i is an atom.

  → corresponds to **formula** in the language of predicate logic.

- an expression is **ground** if it does not contain any variables.

Syntax of Datalog (cont)

definite clause is either an atomic symbol (a fact) or of the form:

$$\underbrace{a}_{\text{head}} \leftarrow \underbrace{b_1 \wedge \cdots \wedge b_m}_{\text{body}}$$

where a and b_i are atomic symbols.

query is of the form $?b_1 \wedge \cdots \wedge b_m$.

knowledge base is a set of definite clauses.

Example Knowledge Base

$in(alan, R) \leftarrow$

$\quad teaches(alan, cs322) \wedge$

$\quad in(cs322, R).$

$grandfather(william, X) \leftarrow$

$\quad father(william, Y) \wedge$

$\quad parent(Y, X).$

$slithy(toves) \leftarrow$

$\quad mimsy \wedge borogroves \wedge$

$\quad outgrabe(mome, Raths).$

Semantics: General Idea

A semantics specifies the meaning of sentences in the language.

An interpretation specifies:

- what objects (individuals) are in the world
- the correspondence between symbols in the computer and objects & relations in world
  - constants denote individuals
  - predicate symbols denote relations

Formal Semantics

An interpretation is a triple $I = \langle D, \phi, \pi \rangle$, where

- D, the domain, is a nonempty set. Elements of D are individuals.
- ϕ is a mapping that assigns to each constant an element of D. Constant c denotes individual $\phi(c)$.
- π is a mapping that assigns to each n-ary predicate symbol a function from D^n into $\{TRUE, FALSE\}$.

Important points to note

- The domain D can contain real objects. (e.g., a person, a room, a course). D can't necessarily be stored in a computer.
- $\pi(p)$ specifies whether the relation denoted by the n-ary predicate symbol p is true or false for each n-tuple of individuals.
- If predicate symbol p has no arguments, then $\pi(p)$ is either *TRUE* or *FALSE*.

Remarks on Semantic Terminology

- Datalog
  - Interpretation $I = < D, \varphi , \pi >$
  - φ denotes objects in the domain.
- Schöning / F2-Vorlesung
  - Structure $A = (U, I)$
  - Universe. Domain: U
  - one interpretation (mapping) I for constants and predicate symbols.
- LOS (Logics and Semantics) course
  - Model $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$

Formalizing the interpretation of figure 2.1

- Domain (entities in the physical world):
  Alan, room 123, room 023, CS–building
- Constants of the language:
  alan, r123, r023, cs_building
- Denotation: φ (alan) = Alan , φ (r123) = room 123 , …
  an abbreviation: φ (alan) = alan´, φ (r123) = r123´
- Predicate symbols: *person, in, part_of*

Formalizing the interpretation of figure 2.1 (cont)

Extension of the predicate symbols:

- π *(person)(*alan´*)* = *TRUE*
  π *(person)(*r123´*)* = *FALSE*
  π *(person)(*r023´*)* = *FALSE*
  π *(person)(*cs_building´*)* = *FALSE*
- π *(in)(*alan´, r123´*)* = *TRUE*
  π *(in)(*alan´, cs_building´*)* = *TRUE*
- π *(part_of)(*r123´, cs_building´*)* = *TRUE*
  π *(part_of)(*alan´, cs_building´*)* = *FALSE*

complete extension (person)

To be extended (in, part_of)

Truth in an interpretation

Each ground term denotes an individual in an interpretation.

A constant c denotes in I the individual $\phi(c)$.

Ground (variable-free) atom $p(t_1, \ldots, t_n)$ is

- true in interpretation I if $\pi(p)(t'_1, \ldots, t'_n) = \mathit{TRUE}$, where t_i denotes t'_i in interpretation I and
- false in interpretation I if $\pi(p)(t'_1, \ldots, t'_n) = \mathit{FALSE}$.

Ground clause $h \leftarrow b_1 \wedge \ldots \wedge b_m$ is false in interpretation I if h is false in I and each b_i is true in I, and is true in interpretation I otherwise.

Models and logical consequences

- A knowledge base, KB, is true in interpretation I if and only if every clause in KB is true in I.
- A model of a set of clauses is an interpretation in which all the clauses are true.
- If KB is a set of clauses and g is a conjunction of atoms, g is a logical consequence of KB, written $KB \models g$, if g is true in every model of KB.
- That is, $KB \models g$ if there is no interpretation in which KB is true and g is false.

Simple Example

$$KB = \begin{cases} p \leftarrow q. \\ q. \\ r \leftarrow s. \end{cases}$$

	$\pi(p)$	$\pi(q)$	$\pi(r)$	$\pi(s)$	
I_1	*TRUE*	*TRUE*	*TRUE*	*TRUE*	is a model of *KB*
I_2	*FALSE*	*FALSE*	*FALSE*	*FALSE*	not a model of *KB*
I_3	*TRUE*	*TRUE*	*FALSE*	*FALSE*	is a model of *KB*
I_4	*TRUE*	*TRUE*	*TRUE*	*FALSE*	is a model of *KB*
I_5	*TRUE*	*TRUE*	*FALSE*	*TRUE*	not a model of *KB*

$KB \models p$, $KB \models q$, $KB \not\models r$, $KB \not\models s$

User's view of Semantics

1. Choose a task domain: intended interpretation.
2. Associate constants with individuals you want to name.
3. For each relation you want to represent, associate a predicate symbol in the language.
4. Tell the system clauses that are true in the intended interpretation: axiomatizing the domain.
5. Ask questions about the intended interpretation.
6. If $KB \models g$, then g must be true in the intended interpretation.

Computer's view of semantics

- The computer doesn't have access to the intended interpretation.
- All it knows is the knowledge base.
- The computer can determine if a formula is a logical consequence of KB.
- If $KB \models g$ then g must be true in the intended interpretation.
- If $KB \not\models g$ then there is a model of KB in which g is false. This could be the intended interpretation.