LEGOLOG

Control and Execution

- Programs: RCX visual language or NQC
  - program controls: sequence, if-then, while, until, etc. with concurrent features (tasks)
- Planning: hard-coded into programs
- Interactions during execution time: none, except the robot's interaction with the environment (sensors)
  - difficult (impossible) to recover from failures
  - no change in the course of actions
  - à inflexibility of robots

High-level Control

- Goals:
  - increase the flexibility of the robot by allowing interactions during execution
  - simplify the robot control by allowing high-level planning execution and monitoring
- Programs: divided into two parts
  - Part 1: Basic actions
    - written in NQC
    - consisting of several basic tasks
    - do not contain the "main" control part that combines the tasks to achieve the goal of the program – the "main" task contains a loop that waits for commands from the controller and executes these commands
  - Part 2: Main control part
    - written in a high-level robot control language called ConGolog
- Execution of programs: controlled by an execution monitoring program called Indigolog

ConGolog – An Overview

- High-level logic programming language
  - primitives: basic actions of a dynamic domain
  - controls:
    - well-known constructs such as sequence, if-then-else, while, procedure
    - special constructs: nondeterministic choice of arguments, nondeterministic choice of actions
    - concurrent constructs
    - prioritized, interrupts

IndiGolog - Overview

- Incremental deterministic ConGolog with sensing actions
- Interpreter:
  - implemented in Eclipse prolog
  - considers exogenous actions
    - interactions during execution

Communication between IndiGolog and RCX

- Originally done through serial port (IR tower), Linux OS:
  - Prolog sends commands to RCX by writing directly to the serial port
  - Prolog reads responses from RCX by reading the serial port
- In this course: we need to change this communication due to the USB port:
  - Two ways connection between Eclipse and JAVA (Eclipse library)
  - Two ways connection between JAVA and RCX (Java SDKMindstorms)

What's next?

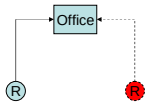
- ConGolog
- Eclipse Prolog
- IndiGolog interpreter
- Communication between Eclipse and Java
- Communication between Java and RCX
- Application: Schedule a meeting (a variation of the delivery example from the Legolog's paper)

ConGolog

- ConGolog programs: primitives are actions of a dynamic domain
- Constructs: well-known program constructs
- Example:
  - go_straight; turn_left
  - if (connected) then go_straight else go_back;
  - while (¬at_intersection) do go_straight
  - while (¬at_goal_line) do go_straight | turn_left | turn_right
  - proc(go_to_room(X),[…])
  - proc(get_coffee,[go_to_room(123), …])
  - ((go_straight; turn_left) | (go_straight; turn_right)); ?(at_office);

ConGolog – Intuition

- Given: a description of the world
  - actions
  - fluents
  - effects of actions on the environment
- ConGolog programs: partial plans



Example:

((go_straight; turn_left) | (go_straight; turn_right)); ?(at_office);

This is a partial plan for the robot to go to the office. At execution time, depending on the current location, the robot will decide which of the programs (go_straight; turn_left) or (go_straight; turn_right) will lead to the goal of being at_office after its execution and then execute it.

Something we have learned …

- Logic
  - Propositional logic
  - First order logic
- Basic definitions
  - Syntax:
    - Term
    - Atom
    - Sentences
  - Semantics:
    - Interpretation
    - Model
    - Satisfiability
    - Entailment
  - Inference, Proof,

Situation Calculus

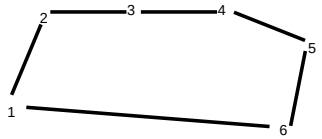
- Multi-sorted first order logic theories with the three sorts: actions, fluents, and situations
- A situation calculus theory is a multi-sorted first order logic theory consisting of
  - the set of actions: Actions, the set of fluents: Fluents
  - function: Do: Actions x Sits à Sits where Sits denotes the set of situations
  - predicate: Holds with two arguments Fluents and Sits
  - for each pair of an action *Act* and a fluent *F*, a *successor state axiom* of the form
    Holds(F, Do(Act,s)) ≡ γ+(Act,F,s) ∨ (Holds(F,s) ∧ ¬ γ-(Act,F,s))
    where γ+(Act,F,s) (γ-(Act,F,s)) denotes the positive (negative) precondition of *Act*
  - a situation constants: S0 – initial situation
  - the set of axioms describing the initial situation, i.e., axioms of the form Holds(F,S0)
  - the set of actions preconditions of the form Poss(act,s) ⊃ γ(act,s)
  - the set of domain-independent foundational axioms
  - the set of domain closure axioms
  - the set of unique name axioms for actions
- A situation term is of the form Do(a_n,Do(a_{n-1},…,Do(a_1,S0))) or S0 (or: Do(act,s) where s is a situation term and act is an action)
- Entailment is defined as in first order logic

Example

Navigation in a graph: a bi-directed graph representing by a set of nodes and a relation *Connected(X,Y)*.
Actions = {*Move(X,Y)* | X, Y are nodes of the graph}
Fluents = {*At(X)* | X is a node of the graph}
S0: Holds(at(0), S0) "initially, the robot is at node 0"
Action precondition:
Poss(Move(X,Y),s) ⊃ Holds(At(X),s) ∧ Connected(X,Y)
Successor state axiom: Holds(At(X),Do(act,s)) ≡ act=Move(Y,X)
Question: What is the truth value of
Holds(At(1), Do(Move(0,1),S0),
Holds(At(0), Do(Move(1,0), Do(Move(0,1), S0))), and
Holds(At(2), Do(Move(1,0), Do(Move(0,1), S0)))
given three nodes 0, 1, 2 and the relation *Connected*={(0,1),(0,2)}.

Another example

The railway domain



1, 2, …, 6: stations – white board
Connections: black tape

Navigation in the railway domain

Actions = {*GoToNextStation(X, 1+X mod 6)* | X is a station}
Fluents = {*At(X)* | X is a station}
S0: Holds(at(1), S0) "initially, the robot is at station 1"
Action precondition:
Poss(Move(X, Y),s) ⊃ Holds(At(X),s) ∧ Y = 1+ X mod 6
Successor state axiom:
Holds(At(X),Do(act,s)) ≡ act=GoToNextStation(Y,X)
Question: What is the truth value of
Holds(At(1), Do(GoToNextStation(6,1), S0),
Holds(At(2), Do(GoToNextStation(1,2), S0), and
Holds(At(3), Do(GoToNextStation(1,2), S0)).

Projection in Situation Calculus

- Given a situation calculus theory D and a sequence of actions $a_1,\ldots a_{n-1},a_n$ and a fluent f.
- Projection is the problem of determining whether

D |= Holds(f, Do(a_n,Do(a_{n-1},…,Do(a_1,S0))))

holds or not.

- Examples: see previous slides