Slide 1

COMP3152/9152
Lecture 7
Knowledge-Based Programs
Ron van der Meyden

Slide 2

A notation for programs

Case of
  if ϕ_1 **do** action$_1$
  ⋮
  if ϕ_n **do** action$_n$
end case

means:
repeat forever: nondeterministically choose i such that ϕ_i is true, do action$_i$

Slide 3

Standard programs

Standard programs for agent i are programs in which the ϕ_i are boolean formulas over atomic propositions "local to agent i"

Example:

Case of
  if ¬ received$_S$(ack,R) **do** send$_S$(m,R)
  if received$_S$(ack,R) **do** skip
end case

Slide 4

A proposition p is *local to agent* i according to an interpretation π over a set of global states $\mathcal{G}$ if for all $s, t \in \mathcal{G}$, if $s \sim_i t$ then $\pi(s)(p) = \pi(t)(p)$.

π is *compatible* with $\mathtt{Pg}_i$ if every proposition p that occurs in $\mathtt{Pg}_i$ is local to i according to π.

If ϕ is a boolean combination of propositions local to i according to π and l is a local state of agent i, define $(\pi, l) \models \phi$ if $(\pi, g) \models \phi$ for all global states g with $g_i = l$.

Slide 5

Knowledge-based programs

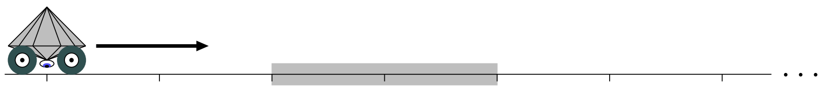
Knowledge-based programs for agent i are programs in which the ϕ may talk about the knowledge of agent i

E.g., After father says "at least one of you is muddy", child i behaves as if running the following knowledge-based program:

Case of
    if $K_i(\mathrm{muddy}_i) \vee K_i(\neg\mathrm{muddy}_i)$ **do** Say "Yes"
    if $\neg(K_i(\mathrm{muddy}_i) \vee K_i(\neg\mathrm{muddy}_i))$ **do** Say "No"
end case

Slide 6

A robotics example



A knowledge-based program:
Case of
    if $K_r(\mathrm{posn} \in \mathrm{Goal})$ **do** `halt`
end case

What is the semantics of such programs? (FHMV book version)

Slide 7

Joint actions

Let ACT_e be a set of actions that the environment can perform

For each agent i, let ACT_i be a set of actions that agent i can perform

(examples: say "yes", send message m to agent j, the "skip/do nothing" action Λ)

A *joint action* is a tuple $(a_e, a_1, \ldots, a_n)$ such that $a_e \in ACT_e$ and $a_i \in ACT_i$ for each $i = 1 \ldots n$.

Write ACT for the set of joint actions

Slide 8

Let the set of global states be $\mathcal{G} = L_e \times L_1 \times \ldots \times L_n$.

A *transition function* is a mapping $\tau : ACT \to (\mathcal{G} \to \mathcal{G})$

if $\mathbf{a}$ is a joint action and s is a state, then $\tau(\mathbf{a})(s)$ is the next state when $\mathbf{a}$ is performed in state s.

Typically, $\tau((\Lambda, \Lambda, \ldots, \Lambda))(s) = s$

Slide 9

Protocols

A *protocol* for agent i (with local states L_i) is a mapping

$$P_i : L_i \to \mathcal{P}(ACT_i) \setminus \{\emptyset\}$$

$P_i(l) = \{a_1, \ldots, a_n\}$ means that when in local state l, agent i's next action is (nondeterministically) one of $a_1, \ldots, a_n$

P_i is *deterministic* if $|P_i(l)| = 1$ for all $l \in L_i$

A *joint protocol* is a tuple $(P_1, \ldots, P_n)$ where each P_i is a protocol of agent $i = 1 \ldots n$

Slide 10

Admissibility Conditions

We sometimes need to express constraints on the set of runs that cannot be captures using just initial states and protocols.

E.g. Every message sent is eventually delivered.

Represent these using a set Ψ of runs, called an *admissibility condition.*

E.g. Ψ is the set of runs in which all messages sent are eventually delivered

Slide 11

Contexts

A *context* is a tuple $\gamma = (P_e, \mathcal{G}_0, \tau, \Psi)$ where

1. $P_e : L_e \to ACT_e$ is a protocol for the environment
2. $\mathcal{G}_0 \subseteq \mathcal{G}$ is a set of *initial global states,*
3. $\tau : ACT \to (\mathcal{G} \to \mathcal{G})$ is a transition function,
4. Ψ is an *admissibility* condition on runs

An *interpreted context* is a pair (γ, π) where π is an interpretation of a set of atomic propositions *Prop* in the global states $\mathcal{G}$ of γ.

Slide 12

A run r is *consistent* with a joint protocol $P = (P_1, \ldots, P_n)$ in context $\gamma = (P_e, \mathcal{G}_0, \tau, \Psi)$ if

1. $r(0) \in \mathcal{G}_0$
2. for all $m \geq 0$ there exists $a_e \in P_e(r_e(m))$ and $a_i \in P_i(r_i(m))$ (for $i = 1 \ldots n$) such that $r(m+1) = \tau((a_e, a_1, \ldots, a_n))(r(m))$
3. $r \in \Psi$

Write $\mathbf{R}^{rep}(P, \gamma)$ for the set of all runs that are consistent with protocol P in context γ

lide 13

Interpreting Standard Programs as Protocols

Let (γ, π) be an interpreted context and

$\mathtt{Pg}_i$ = **Case of**

if ϕ_1 **do** a_1

$\vdots$

if ϕ_n **do** a_n

end case

a standard program for agent i such that π is compatible with $\mathtt{Pg}_i$.

Define the protocol $\mathtt{Pg}_i^\pi$ by

$$\mathtt{Pg}_i^\pi(l) = \begin{cases} \{a_i \mid (\pi, l) \models \phi_i, i = 1 \ldots n\} & \text{if this set is not } \emptyset \\ \{\Lambda\} & \text{otherwise} \end{cases}$$

lide 14

For joint programs $\mathtt{Pg} = (\mathtt{Pg}_1, \ldots, \mathtt{Pg}_n)$,

$$\mathtt{Pg}^\pi = (\mathtt{Pg}_1^\pi, \ldots, \mathtt{Pg}_n^\pi)$$

The *interpreted system representing* a joint program $\mathtt{Pg}$ in the interpreted context (γ, π) is the system $\mathcal{I}^{rep}(\mathtt{Pg}, \gamma, \pi) = (\mathbf{R}^{rep}(\mathtt{Pg}^\pi, \gamma), \pi)$

Slide 15

Can we give a similar definition $\mathtt{Pg}^\pi$ for knowledge-based programs?

A problem:

to determine the set of runs produced by executing a knowledge based program, we need to know which actions are enabled at each point (r, m).

to decide whether an action a_i is enabled by a knowledge-based program at (r, m), we need to evaluate the formula ϕ_i.

When ϕ_i contains knowledge operators, this means looking at points (r', m') such that $(r', m') \sim_i (r, m)$

For that, we need to know the set of runs.

This is circular!

Slide 16

Resolution: provide a way of *testing* whether a given standard program/protocol implements a knowledge-based program.....

Given an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ and a joint knowledge-based program $\mathtt{Pg} = (\mathtt{Pg}_1, \ldots, \mathtt{Pg}_n)$, define a joint protocol $\mathtt{Pg}^\mathcal{I} = (\mathtt{Pg}_1^\mathcal{I}, \ldots, \mathtt{Pg}_n^\mathcal{I})$, as follows.

lide 17

A knowledge-based formula ϕ is *local to agent i according to* π if it is either a proposition p local to i according to π, a formula of the form $K_i\phi$, or a boolean combination of these

Let $\mathcal{I} = (\mathcal{R}, \pi)$ be an interpreted system. If l is a local state of agent i and ϕ is a formula local to i according to π, define $(\mathcal{I}, l) \models \phi$ by

1. $(\mathcal{I}, l) \models p$ if $(\pi, l) \models p$ (as defined above),
2. $(\mathcal{I}, l) \models K_i\phi$ if $(I, r, m) \models \phi$ for all points (r, m) in $\mathcal{I}$ such that $r_i(m) = l$,
3. (Booleans as usual)

lide 18

Let $\mathcal{I} = (\mathcal{R}, \pi)$ be an interpreted system and
$\mathtt{Pg}_i$ = **Case of**
    if ϕ_1 **do** a_1
    $\vdots$
    if ϕ_n **do** a_n
end case
a knowledge-based program for agent i such that each ϕ_j is local to i according to π.

Define the protocol $\mathtt{Pg}_i^{\mathcal{I}}$ by

$$\mathtt{Pg}_i^{\mathcal{I}}(l) = \begin{cases} \{a_i \mid (\mathcal{I}, l) \models \phi_i, i = 1 \ldots n\} & \text{if this set is not } \emptyset \\ \{\Lambda\} & \text{otherwise} \end{cases}$$

Slide 19

A system $\mathcal{I}$ *represents* the knowledge-based program $\mathtt{Pg}$ in an interpreted context (γ, π) if π is compatible with $\mathtt{Pg}$ and $\mathcal{I} = \mathcal{I}^{rep}(\mathtt{Pg}^{\mathcal{I}}, \gamma, \pi)$.

A standard protocol P is an *implementation* of the knowledge-based program $\mathtt{Pg}$ in an interpreted context (γ, π) if $\mathcal{I}_P = \mathcal{I}^{rep}(P, \gamma, \pi)$ represents $\mathtt{Pg}$ in (γ, π).

Note this implies $\mathcal{I}_P = \mathcal{I}^{rep}(\mathtt{Pg}^{\mathcal{I}_P}, \gamma, \pi)$.

A standard program $\mathtt{Pg}_s$ is an *implementation* of the knowledge-based program $\mathtt{Pg}_k$ in an interpreted context (γ, π) if the protocol $\mathtt{Pg}_s^{\pi}$ is an implementation of $\mathtt{Pg}_k$ in (γ, π).

Slide 20

Specifications

Let ϕ be a formula of the language of knowledge and time

A protocol P satisfies ϕ in context (γ, π) if for all runs $r \in \mathbf{R}^{rep}(P, \gamma)$, we have

$$((\mathbf{R}^{rep}(P, \gamma), \pi), r, 0) \models \phi \quad .$$

A (knowledge-based) program $\mathtt{Pg}$ satisfies ϕ in (γ, π) if for all systems $\mathcal{I}$ representing $\mathtt{Pg}$ in (γ, π), for all runs r of $\mathcal{I}$ we have

$$(\mathcal{I}, r, 0) \models \phi \quad .$$

Slide 21

Specification of Robot

$\texttt{Goal} = \{2, 3, 4\}$

Safety part: $\Box(\texttt{halted} \Rightarrow \texttt{posn} \in \texttt{Goal})$

Liveness part: $\Diamond\texttt{halted}$

Slide 22

Implementations of the robot example

Let P^Λ be the deterministic protocol that always does Λ (never `halt`)

Let γ be a context in which

- the environment nondeterministically moves the position one step, or none
- the robot's local state is the sensor reading
- the sensor reading is (nondeterministically) $\texttt{posn} + x$, $x \in \{-1, 0, 1\}$

Then

$\mathcal{I}^{rep}(P^\Lambda, \gamma, \pi) \models K_r(\texttt{posn} \in \{2, 3, 4\}) \iff \texttt{sensor} = 3)$

Slide 23

Let MP be the knowledge-based program

if $K_r(\texttt{posn} \in \{2, 3, 4\})$ **do** `halt`

Let MP_s be the standard program

if `sensor=3` **do** `halt`

Then MP_s is an implementation of MP in (γ, π)

MP_s satisfies the specification $\Box(\texttt{halted} \Rightarrow \texttt{posn} \in \{2, 3, 4\})$

But NOT the liveness part: $\Diamond\texttt{halted}$

Slide 24

Let MP'_s be the standard program

if $\texttt{sensor} \in \{3, 4, 5\}$ **do** `halt`

Then MP_s is *also* an implementation of MP in (γ, π)

MP_s satisfies the safety specification $\Box(\texttt{halted} \Rightarrow \texttt{posn} \in \{2, 3, 4\})$

AND the liveness part: $\Diamond\texttt{halted}$

lide 25

A KBP with NO implementations

There are NO implementations of the KBP

if $K_1(\neg\Diamond(bit = 1))$ **do** $bit := 1$

in a context where the bit is initially 0, agent 1's local state is the value of *bit*, and the environment does nothing

lide 26

Unique Implementations

The above examples show that we can have 0,1 or many implementations of a knowledge-based program.

(So knowledge-based programs are more like specifications than like programs.)

When can we be sure there is a *unique* implementation?

Slide 27

Nonexcluding Admissibility Conditions

The admissibility condition Ψ of a context $\gamma = (P_e, \mathcal{G}_0, \tau, \Psi)$ is *nonexcluding* if for every protocol P and all times m, if $r : [0, \ldots m] \to \mathcal{G}$ satisfies

1. $r(0) \in \mathcal{G}_0$ and
2. for all $k \in [0, \ldots, m-1]$ and $(a_e, a_1, \ldots, a_n) \in P_e(r(k)) \times P_1(r_1(k)) \times \ldots \times P_n(r_n(k))$ we have $r(k+1) = \tau((a_e, a_1, \ldots, a_n))(r(k))$,

then there exists a run $r' \in \mathbf{R}^{rep}(P, \gamma)$ such that $r[0 \ldots m] = r'[0 \ldots m]$.

Slide 28

An interpreted system $\mathcal{I}$ provides witnesses for $K_i\phi$ if for every point (r, m) of $\mathcal{I}$, if $(\mathcal{I}, r, m) \models \neg K_i\phi$ then there exists $m' \leq m$ such that $(r, m) \sim_i (r', m')$ and $(\mathcal{I}, r', m') \models \neg\phi$.

Lemma: Every synchronous system provides witnesses for every formula $K_i\phi$.

lide 29

An interpreted context (γ, π) provides witnesses for a knowledge-based program Pg if every interpreted system $\mathcal{I}$ representing Pg in (γ, π) provides witnesses for every formula $K_i\phi$ that appears as a subformula of a test in Pg.

lide 30

Theorem: Let Pg be a knowledge-based program in which tests do not involve temporal operators, let γ be a nonexcluding context, and assume that the context (γ, π) provides witnesses for Pg. Then there is a unique interpreted system representing Pg in (γ, π).

(There may still be many protocols/programs implementing Pg, but these differ only on unreachable states and syntactic details.)