Model Checking Knowledge-based Programs

Alexander Knapp
Heribert Mühlberger
Universität Augsburg

Sum and Product

“J says to S and P: I have chosen two integers x and y such that $1 < x \leq y < 100$. In a moment, I will inform S only of $s = x + y$, and P only of $p = x \cdot y$. These announcements remain private. You are required to determine the pair (x, y). He acts as said. The following conversation now takes place:

1. P says: ‘I do not know it.’
2. S says: ‘I knew you didn’t.’
3. P says: ‘I now know it.’
4. S says: ‘I now also know it.’

Determine the pair (x, y).”

H. Freudenthal (1969)

Epistemic Logic

$\varphi, \psi \in \mathcal{L}^n_{\mathrm{K}}(P) ::= p \mid \neg\varphi \mid \varphi \vee \psi \mid \mathrm{K}_i\varphi$

- set of propositions $P \ni p$, finite set of agents $\{1, \ldots, n\} \ni i$
- $\mathrm{K}_i\varphi$ read as "agent i knows φ"

Interpreted over Kripke structure $\mathscr{M} = (S, (R_i)_{1 \leq i \leq n}, \pi)$

- set of states S, interpretation of propositions $\pi : S \to \wp P$
- accessibility relation $R_i \subseteq S \times S$ of possible worlds for each agent i

Satisfaction relation $\mathscr{M}, s \models \varphi$

$$\mathscr{M}, s \models p \iff p \in \pi(s)$$
$$\mathscr{M}, s \models \neg\varphi \iff \text{not } \mathscr{M}, s \models \varphi$$
$$\mathscr{M}, s \models \varphi \vee \psi \iff \mathscr{M}, s \models \varphi \text{ or } \mathscr{M}, s \models \psi$$
$$\mathscr{M}, s \models \mathrm{K}_i\varphi \iff \text{for all } t \in S \text{ with } (s, t) \in R_i\text{: } \mathscr{M}, t \models \varphi$$

Epistemic Logic: Axioms

$\mathbf{S5}_n$-logic — all R_i equivalence relations

- $(\mathrm{K}_i\varphi \wedge \mathrm{K}_i(\varphi \rightarrow \psi)) \rightarrow \mathrm{K}_i\psi$ — distribution axiom
- $\mathrm{K}_i\varphi \rightarrow \varphi$ — knowledge axiom
  - "Known facts are true"
  - R_i reflexive on S
- $\mathrm{K}_i\varphi \rightarrow \mathrm{K}_i\mathrm{K}_i\varphi$ — positive introspection axiom
  - "If agent i knows φ, then he knows that he knows φ"
  - R_i transitive
- $\neg\mathrm{K}_i\varphi \rightarrow \mathrm{K}_i\neg\mathrm{K}_i\varphi$ — negative introspection axiom
  - "If agent i does not know φ, then he knows that he does not know φ"
  - R_i Euclidean
- $\neg\mathrm{K}_i\mathrm{false}$ — consistency axiom
  - "No agent believes false"
  - R_i serial

Knowledge in Programs

Programs with knowledge guards

- abstracting from how knowledge is gained

Bit-transmission protocol

- $\texttt{if}\ \neg K_{\text{Sender}}\,\text{recbit}\ \texttt{then}\ \text{sendbit}$
- $\texttt{if}\ K_{\text{Receiver}}\,\text{bit} \land \neg K_{\text{Receiver}}\,K_{\text{Sender}}\,K_{\text{Receiver}}\,\text{bit}\ \texttt{then}\ \text{sendack}$

Sum-and-product

- $\texttt{if}\ \text{step} = 1 \land K_S(\neg\exists a \in [2..99]\,.\,K_P\,x = a)\ \texttt{then}\ \text{step} \leftarrow \text{step} + 1$

Based on R. Fagin, J. Y. Halpern, Y. Moses, M. Y. Vardi (1995)

Knowledge-based Programs

Finite set of propositions P

- determines set of states $\wp P \ni s$

Observability set $W \subseteq P$

- an agent can observe propositions in W
- defines equivalence relation $s_1 \sim_W s_2 \iff$ for all $p \in P$: $p \in s_1$ iff $p \in s_2$

Knowledge-based program $(T, (W_i)_{1 \leq i \leq n}, \gamma, I)$ over P

- Transition relation $T \subseteq \wp P \times \wp P$
- Observability set for each agent $i \in \{1, \ldots, n\}$
- Assignment of knowledge guards $\gamma : T \to \mathcal{L}^n_{\mathrm{K}}(P)$
- Initial states $I \subseteq \wp P$

Knowledge-based Programs: Sum-and-Product (1)

```
specification sum_and_product;

var x, y : 2..99 initial x <= y;
var s : 4..198 initial s = x + y;
var p : 4..9801 initial p = x * y;
var step : 1..6 initial step = 1;
var p1, p2, p3, p4, s2: boolean
      initial p1 = false & p2 = false & p3 = false &
              p4 = false & s2 = false;

agent Prod = {    p, step, p1, p2, p3, p4     };
agent Sum =  { s,    step, p1, p2, p3, p4, s2 };

guard P_knows_x = (exists a:2..99 . (K[Prod] x = a));
guard P_knows_y = (exists b:2..99 . (K[Prod] y = b));
guard S_knows_x = (exists a:2..99 . (K[Sum] x = a));
guard S_knows_y = (exists b:2..99 . (K[Sum] y = b));
guard S_knows_P_does_not_know_x =
        K[Sum] ~(exists a:2..99 . (K[Prod] x = a));
```

Knowledge-based Programs: Sum-and-Product (2)

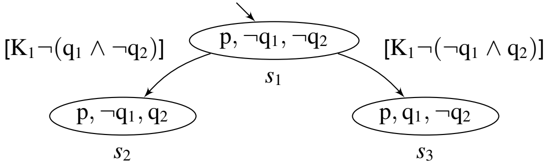
```
action step1_S_yes
epre S_knows_P_does_not_know_x
pre  step = 1
do s2 := true, step := step + 1;

action step1_S_no
epre ~S_knows_P_does_not_know_x
pre  step = 1
do s2 := false, step := step + 1;

action step2_P_yes
epre P_knows_x
pre  step = 2
do p1 := true, step := step + 1;

action step2_P_no
epre ~P_knows_x
pre  step = 2
do p1 := false, step := step + 1;

action step3_S_publish
pre step = 3
do p2 := s2, step := step + 1;
```

Knowledge-based Programs: Sum-and-Product (3)

```
action step4_P_yes
epre P_knows_x
pre  step = 4
do p3 := true, step := step + 1;

action step4_P_no
epre ~P_knows_x
pre  step = 4
do p3 := false, step := step + 1;

action step5_S_yes
epre S_knows_x
pre  step = 5
do p4 := true, step := step + 1;

action step5_S_no
epre ~S_knows_x
pre  step = 5
do p4 := false, step := step + 1;

action stutter
pre step = 6
do ;

end;
```

Knowledge-based Programs: Interpretation

Propositions $P = \{\mathrm{p}, \mathrm{q}_1, \mathrm{q}_2\}$, observability set $W_1 = \{\mathrm{p}\}$ for agent 1



- Possible runs depend on evaluation of knowledge guards
- Evaluation of knowledge guards depends on possible runs
  - Which states are reachable and therefore possible worlds?

Interpreting Knowledge-based Programs

Knowledge-based program $\mathsf{S} = (T, (W_i)_{1 \leq i \leq n}, \gamma, I)$ over P

Interpretation of S w.r.t. possible worlds $S \subseteq \wp P$

Kripke structure $\mathscr{M}(\mathsf{S}, S) = (S, (R_i)_{1 \leq i \leq n}, \pi)$ for $S \subseteq \wp P$

- $R_i = \sim_{W_i} \cap (S \times S)$
- $\pi(s) = s$

Evaluation of knowledge guards of S w.r.t. $S \subseteq \wp P$ and $s \in \wp P$

$$\mathsf{S}, S, s \models \mathrm{K}_i \varphi \iff \text{for all } s' \in S \text{ with } s \sim_{W_i} s' \colon \mathscr{M}(\mathsf{S}, S), s' \models \varphi$$

Reachable states $\mathcal{R}_{\mathsf{S}}(S) \subseteq \wp P$ of S w.r.t. possible worlds S

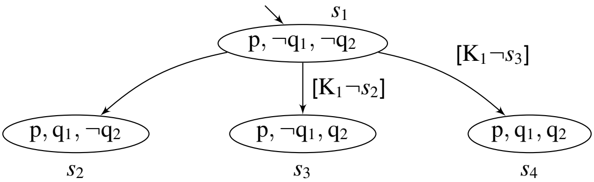
- compute reachable states w.r.t. S by evaluating knowledge guards η in a state s with $\mathsf{S}, S, s \models \eta$

Goal: Unique interpretation with $\mathcal{R}_{\mathsf{S}}(S) = S$

Unique Interpretation (1)

Propositions $P = \{\mathrm{p}, \mathrm{q}_1, \mathrm{q}_2\}$, observability set $W_1 = \{\mathrm{p}\}$ for agent 1

- abbreviating valuation of propositions by state name

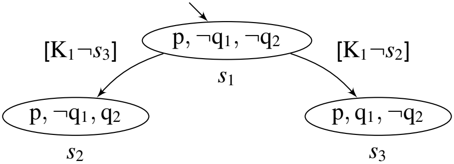


$$\mathcal{R}_{\mathrm{S}}(\emptyset) = \{s_1, s_2, s_3, s_4\}$$
$$\mathcal{R}_{\mathrm{S}}(\{s_1, s_2, s_3, s_4\}) = \{s_1, s_2\}$$
$$\mathcal{R}_{\mathrm{S}}(\{s_1, s_2\}) = \{s_1, s_2, s_4\}$$
$$\mathcal{R}_{\mathrm{S}}(\{s_1, s_2, s_4\}) = \{s_1, s_2, s_4\}$$

Not monotone

Unique Interpretation (2)

Propositions $P = \{\mathrm{p}, \mathrm{q}_1, \mathrm{q}_2\}$, observability set $W_1 = \{\mathrm{p}\}$ for agent 1



$$\mathcal{R}_\mathsf{S}(\emptyset) = \{s_1, s_2, s_3\}$$
$$\mathcal{R}_\mathsf{S}(\{s_1, s_2, s_3\}) = \{s_1\}$$
$$\mathcal{R}_\mathsf{S}(\{s_1\}) = \{s_1, s_2, s_3\}$$

$$\mathcal{R}_\mathsf{S}(\{s_1, s_2\}) = \{s_1, s_2\}$$
$$\mathcal{R}_\mathsf{S}(\{s_1, s_3\}) = \{s_1, s_3\}$$

Several fixed points

Perfect Synchrony

A system works in perfect synchrony if all reactions of the system are executed in 0-time: all outputs are generated at the same instant of time at which the inputs are present.

Based on logical time

- computation separated into macro steps for interactions with the system
- each macro step consists of a finite number of micro steps for computing the reaction, taking 0-time

Realised in Esterel (J.-P. Marmorat, J.-P. Rigault, G. Berry 1980s)

- based on signals with status in a macro step: present or absent

Esterel: Example

```
module P1:
  input I;  output O;
  signal S1, S2 in
    present I then emit S1 end
  ||
    present S1 else emit S2 end
  ||
    present S2 then emit O end
  end signal
end module
```

Logical coherence — A signal s is present in a macro step iff an `emit` s is executed in this macro step

Logical correctness — For each signal in each macro step there is a unique status (present/absent) such that logical coherence is satisfied

- there is at least one program execution: logically reactive
- there is at most one program execution: logically determined

Esterel: Logical Correctness

```
module P3:
  present O else emit O end
end module
```

- Not logically correct: non-reactive

```
module P4:
  present O then emit O end
end module
```

- Not logically correct: non-determined

```
module P8:
  present O1 then emit O1 end
||
  present O1 then
    present O2 else emit O2 end
  end
end module
```

- Logically correct (combines `P3` and `P4`)

Esterel: Constructive Semantics

Analysis what a statement must do and cannot do

- based on a logical operational semantics
- no checking of assumptions of status of signals

Restriction of logical coherence to constructive coherence

- A signal s is present in a macro step iff an `emit` s must be executed in this macro step.
- A signal s is absent in a macro step iff an `emit` s cannot be executed in this macro step.

Esterel: Must- and Cannot-Analysis

out$(P, I) \equiv$
$E \leftarrow I \cup \{s^\perp \mid s \in \text{outdecls}(P)\}$
do
$E' \leftarrow E$
$C \leftarrow \text{can}^+_S(P, E)$
$M \leftarrow \text{must}_S(P, E)$
$E \leftarrow I \cup \{s^+ \mid s \in M\} \cup \{s^- \mid s \in \text{outdecls}(P) \setminus C\} \cup \{s^\perp \mid s \in C \setminus M\}$
while $E \neq E'$
if $\exists s \,.\, s^\perp \in E$ **then** error(“not constructive”)
return E

- $P =$ `emit S; present S then emit O else pause end`, $I = \emptyset$, outdecls$(P) = \{\mathtt{S}, \mathtt{O}\}$
- $\text{can}^+_S(P, \{\mathtt{S}^\perp, \mathtt{O}^\perp\}) = \{\mathtt{S}, \mathtt{O}\}$
- $\text{must}_S(P, \{\mathtt{S}^\perp, \mathtt{O}^\perp\}) = \{\mathtt{S}\}$
- $\text{must}_S(P, \{\mathtt{S}^+, \mathtt{O}^\perp\}) = \{\mathtt{S}, \mathtt{O}\}$

Re-interpreting Knowledge-based Programs

Application of must/cannot-analysis to interpretation of knowledge-based program S

- Assume two disjoint sets of states:
  M — definitely reachable (positive, must) and
  N — definitely not reachable (negative, cannot)
- Evaluation of knowledge guards of S w.r.t. (M, N)
  S, $(M, N), s \models_{\mathrm{p}} \eta$
  S, $(M, N), s \models_{\mathrm{n}} \eta$
- Compute new pair $(M', N') = \mathcal{R}^{\mathrm{PN}}_{\mathsf{S}}(M, N)$
  M' — reachable states using S, $(M, N), s \models_{\mathrm{p}} \eta$
  N' — complement of reachable states using S, $(M, N), s \not\models_{\mathrm{n}} \eta$

Goal: (unique) interpretation $\mathcal{R}^{\mathrm{PN}}(M, N) = (M, N)$ such that each state either is in M or N

Positive-Negative-Semantics

$$\mathsf{S}, (M,N), s \models_{\mathrm{p}} p \iff p \in s$$
$$\mathsf{S}, (M,N), s \models_{\mathrm{n}} p \iff p \notin s$$

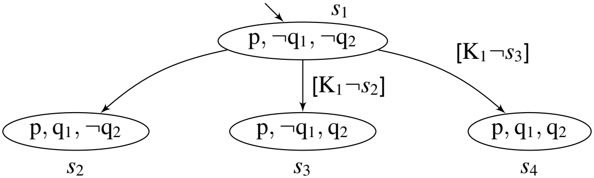
$$\mathsf{S}, (M,N), s \models_{\mathrm{p}} \neg\varphi \iff \mathsf{S}, (M,N), s \models_{\mathrm{n}} \varphi$$
$$\mathsf{S}, (M,N), s \models_{\mathrm{n}} \neg\varphi \iff \mathsf{S}, (M,N), s \models_{\mathrm{p}} \varphi$$

$$\mathsf{S}, (M,N), s \models_{\mathrm{p}} \varphi \vee \psi \iff \mathsf{S}, (M,N), s \models_{\mathrm{p}} \varphi \text{ or } \mathsf{S}, (M,N), s \models_{\mathrm{p}} \psi$$
$$\mathsf{S}, (M,N), s \models_{\mathrm{n}} \varphi \vee \psi \iff \mathsf{S}, (M,N), s \models_{\mathrm{n}} \varphi \text{ and } \mathsf{S}, (M,N), s \models_{\mathrm{n}} \psi$$

$$\mathsf{S}, (M,N), s \models_{\mathrm{p}} \mathrm{K}_i\varphi \iff \text{for all } s' \in [s]_{\sim_i} \text{ with } \mathsf{S}, (M,N), s' \not\models_{\mathrm{p}} \varphi\text{: } s' \in N$$
$$\mathsf{S}, (M,N), s \models_{\mathrm{n}} \mathrm{K}_i\varphi \iff \text{exists } s' \in P \cap [s]_{\sim_i} \text{ such that } \mathsf{S}, (M,N), s' \models_{\mathrm{n}} \varphi$$

Unique Interpretation with Positive-Negative-Semantics (1)

Propositions $P = \{\mathrm{p}, \mathrm{q}_1, \mathrm{q}_2\}$, observability set $W_1 = \{\mathrm{p}\}$ for agent 1

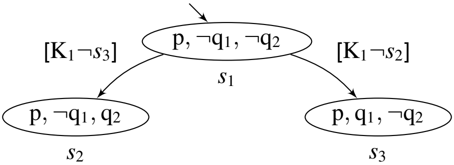


$$\mathcal{R}^{\mathrm{PN}}_{\mathsf{S}}(\emptyset, \emptyset) = (\{s_1, s_2\}, \emptyset)$$
$$\mathcal{R}^{\mathrm{PN}}_{\mathsf{S}}(\{s_1, s_2\}, \emptyset) = (\{s_1, s_2\}, \{s_3\})$$
$$\mathcal{R}^{\mathrm{PN}}_{\mathsf{S}}(\{s_1, s_2\}, \{s_3\}) = (\{s_1, s_2, s_4\}, \{s_3\})$$
$$\mathcal{R}^{\mathrm{PN}}_{\mathsf{S}}(\{s_1, s_2, s_4\}, \{s_3\}) = (\{s_1, s_2, s_4\}, \{s_3\})$$

Monotone

Unique Interpretation with Positive-Negative-Semantics (2)

Propositions $P = \{\mathrm{p}, \mathrm{q}_1, \mathrm{q}_2\}$, observability set $W_1 = \{\mathrm{p}\}$ for agent 1



$$\mathcal{R}^{\mathrm{PN}}_{\mathsf{S}}(\emptyset, \emptyset) = (\{s_1\}, \emptyset)$$
$$\mathcal{R}^{\mathrm{PN}}_{\mathsf{S}}(\{s_1\}, \emptyset) = (\{s_1\}, \emptyset)$$

Undecisive fixed point

Conclusions and Future Work

Model checking approach to knowledge-based programs

- extending MCK (P. Gammie, R. van der Meyden 2004), MCMAS (A. Lomuscio, F. Raimondi 2006), MCTK (X. Luo et al. 2008)
- Alternative: Dynamic Epistemic Logic, DEMO (H. P. van Ditmarsch et al. 2005)

Possible applications

- Security protocols
- Java memory model