Hintikka's world.

**Goals:**

1. Promoting the tool HW
2. Epistemic logic in CS
   a) Knowledge-based programs.
   b) Asynchrony

1) Tool HW

a) History

<table>
<thead>
<tr>
<th>Baruti's world</th>
<th>FO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Kripke's world</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tableau prover</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hintikka's world</th>
<th>DEL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b) Practical review

google Hintikka's world.
1) Dynamic epistemic logic (behind the scenes)

- epistemic situation
- Kripke model
- events
- event model

Diagram:

- Node 1: \( u \) with \( \neg e \) and \( \neg \neg e \): ~
  - Action 1: \( m_r \): \( p \)
  - Action 2: \( m_l \): \( \neg p \)

- Node 2: \( e' \)
  - Action 1: \( m_r \): \( p \)
  - Action 2: \( m_l \): \( \neg p \)
d) Model checking

$\forall \phi, w \models \psi$

without $\ast$, e.g., $\forall \phi, w \models K_a[\phi, w][\phi', w'] \Rightarrow K_b \psi$

$\Rightarrow$ decidable

$\Rightarrow$ PSPACE-complete even if models are symbolic

with $\ast$: e.g., $\forall \phi, w \models [((\phi, \chi) \cup (\phi', \chi'))^\ast] \Rightarrow K_b \psi$

$\Rightarrow$ undecidable.
2) Epistemic logic in CS

a) Knowledge-based programs

Ex:

```
A
if (Kap)
  A
else
  B
while 7Ka9
  C
```

→ easy to read by humans
→ explainable AI

Ex of use in the future:

```
model of
MAS
Qdec-POMDP
```

planer

KB1 for each agent

[AAAI 2018]
Goal: define a semantics for KBPs.

\[ \text{Qdec POMDP) } \vdash \text{ Knight's models.} \]

\[ \text{Def: } (\text{Qdec POMDP}) \text{ a \& transition system } \mathcal{T} \]

\[ \text{with transition of the form:} \]

\[ \text{state} s \xrightarrow{a_{t_1}, a_{t_2}, \text{obs}_1, \text{obs}_2} \text{state } s' \]

\[ \text{e.g. turn right, left; there is a wall} \]

and \[ \text{with an } \theta \text{ set } 0^\circ \text{ of initial states.} \]

\[ \text{Def: } (\mathcal{M}_0) \text{ Given } \mathcal{T}, (\mathcal{M}_0) \text{ where, } \mathcal{M}_0 \text{ is} \]

\[ \text{Worlds of } \mathcal{M}_0: \text{ } 0^\circ \text{ where } 0^\circ \in \theta \]

\[ 0^\circ \text{ is a vector of program contents initial} \]

\[ \mathcal{M}_0: \text{ complete relations over worlds.} \]
**Def:** \( \mathcal{M}_{t+1} \) from \( \mathcal{M}_t \) by induction on \( t \).

**Worlds of** \( \mathcal{M}_{t+1} \) **are of the form:**

- History of depth \( t \) gives the evaluation.
- Record of \( \mathcal{M}_t \) \( s, \sigma \), \( \tau \).
- \( w_t \), \( s^{t+1} \), \( c^{t+1} \):
  - \( \text{success of } c_t \).
  - \( \text{next outcome of } w_t \) in \( c^{t+1} \).

Should satisfy the guard of \( c_t \).

\[ \text{if} \]
1. \( h^t \), \( w_t \rightarrow_t^1 h^t \)
2. \( w^t \rightarrow_t^1 w_t' \)
Def: (program counter)
<guard, action, continuation>.

Ex: [Diagram]

if (KB will)
    KB will
else
    1 Kg will

Ex: [Flowchart]

Verif: of while-free programs: PSPACE-complete
b) Asynchrony

**Architecture:**

\[ q_1, q_2, \ldots, q_n \]

\[ \vdash \quad \vdash \]

\[ \Rightarrow \quad \text{demo} \]

**Idea:** \( (n, q, c) \vdash (n, q', c') \) if

\[ (n, q, c) \vdash (n, q', c') \]

and a has received the same message in the same order as \( m_n \) and \( m_n \).

\[ c[i] = c[i] \land q'[1..c[i]] = q'[1..c[i]] \]

**Bad news:** epidemic merges +

merges are true when sent +

asynchrony

\[ \Rightarrow \text{circularity in the truth condition def.} \]
Example:

\[ m, \Gamma \vdash \text{Kap} ? \]

2a

\[ m \models \text{Kap} \]

\[ m \models \vdash \text{Kap} ? \]

**Conclusion - future work**

**Tools**

1) Design

2) Symbolic models

3) Extensions

**Research**

1) Face undecidability of verified
   \[ \rightarrow \text{loaded plans} \rightarrow \text{approx.} \]

2) Strategic reasoning = KBPs.

3) Face the similarity gap due to...