Towards a Unifying Logic-Based Framework for

Programming

Databases

AI knowledge representation and problem-solving

Robert Kowalski and Fariba Sadri

Department of Computing
Imperial College London

Outline

- KELPS - a simplified kernel for reactive logic-based production-style systems
- Related work (MetateM, Transaction Logic)
- LPS = KELPS + Logic Programs
- MALPS = Multi Agent LPS (The Dining Philosophers)
- Model-theoretic semantics
- Operational semantics
- The frame problem
- Conclusions

KELPS - a simplified kernel for reactive logic-based production-style systems

Programs are reactive rules with explicit time
in the logical form:

$$\forall X\ [antecedent(X) \rightarrow \exists Y\ consequent(X, Y)]$$

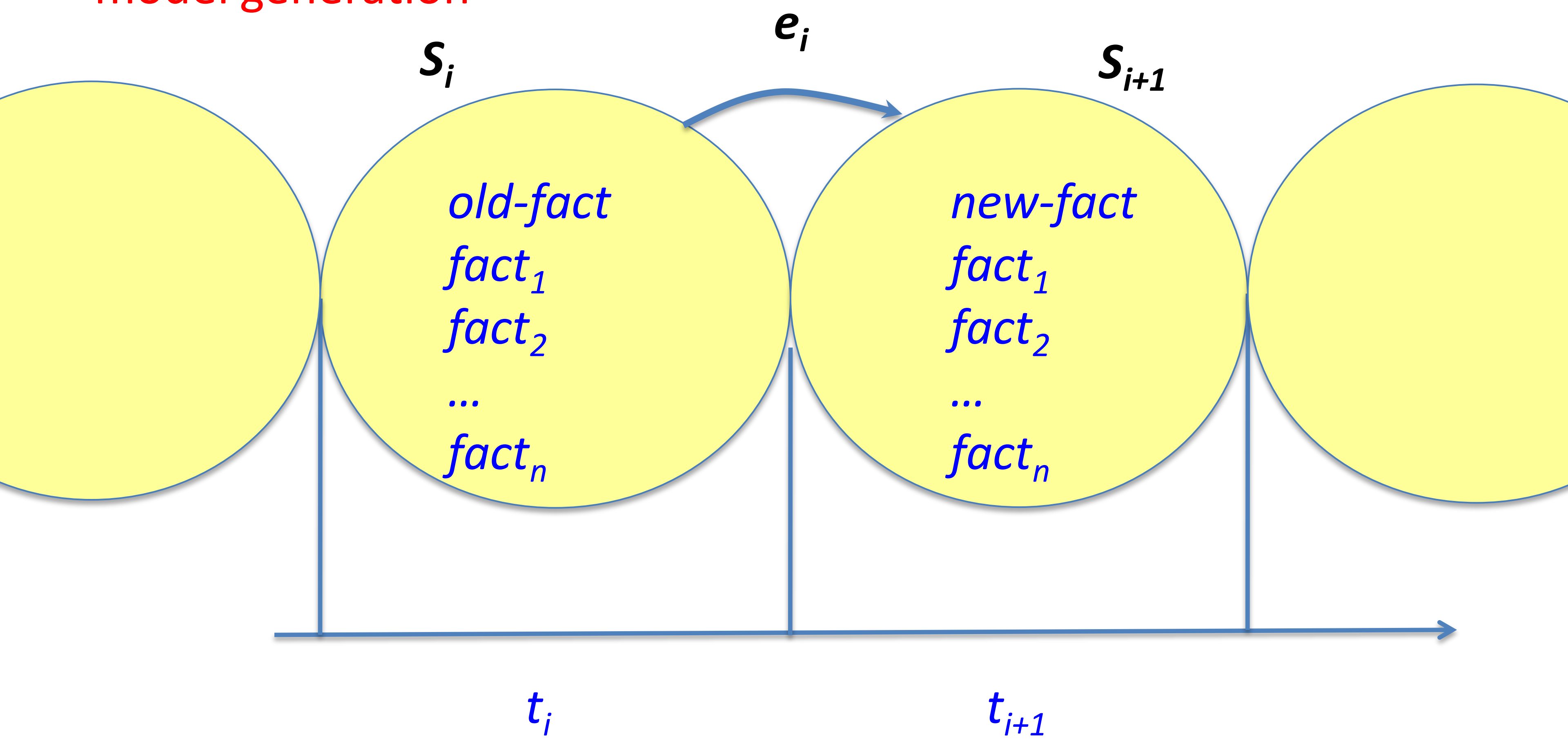
abbreviated $\quad antecedent(X) \rightarrow consequent(X, Y)$

whenever the *antecedent* is true,
then the *consequent* is true in the future.

Model-theoretic semantics:
facts are time-stamped

Operational Semantics:
facts updated destructively
without time stamps

Computation:
model generation



KELPS = Composite events + rules + composite actions

pre-sensor detects possible fire in area A at time $T_1 \wedge$
smoke detector detects smoke in area A at time $T_2 \wedge$
$|T_1 - T_2| \leq 60\ sec \wedge max(T_1, T_2, T)$

$\rightarrow$ *activate local fire suppression in area A at time* $T_3 \wedge T < T_3 \leq T + 10\ sec \wedge$
send security guard to area A at time $T_4 \wedge T_3 < T_4 \leq T_3 + 30\ sec$

$\vee$ *call fire department to area A at time* T_3' $\wedge T < T_3' \leq T + 120\ sec$

Syntax of reactive rules in KELPS

$$antecedent_1(X) \wedge \cdots \wedge antecedent_n(X)$$

$$\rightarrow$$

$$consequent_{11}(X, Y) \wedge \cdots \wedge consequent_{1l1}(X, Y)$$

$$\vee \cdots$$

$$\vee\ consequent_{m1}(X, Y) \wedge \cdots \wedge consequent_{mlm}(X, Y)$$

Each $antecedent_i(X)$ and $consequent_{i\,j}(X, Y)$ is:

- an FOL condition in the vocabulary of state predicates (operationally a query to the extended current state)
- an *event atom* representing an event (including action)
- a *temporal constraint* $time_1 < time_2$ or $time_1 \leq time_2$

Each $consequent_{i1}(X, Y) \wedge \cdots \wedge consequent_{il1}(X, Y)$ is a plan.

Outline

- KELPS
- Related work (MetateM, Transaction Logic)
- LPS = KELPS + Logic Programs
- MALPS = Multi Agent LPS (The Dining Philosophers)
- Model-theoretic semantics
- Operational semantics
- The frame problem
- Conclusions

MetateM (Michael Fisher et al)

Programs = reactive rules in modal temporal logic:

‘past and present formula’ **implies** ‘present or future formula’

Computation = model generation.

Model = possible worlds connected by an accessibility relation.

States updated non-destructively by frame axioms.

Transaction Logic (Bonner and Kifer)

Programs = sequences of FOL queries and database updates.

Computation = model generation.

Model = possible worlds.
Truth defined relative to paths between possible worlds.

States (databases) updated destructively.

Outline

- KELPS
- Related work
- LPS = KELPS + Logic Programs
- MALPS = Multi Agent LPS (The Dining Philosophers)
- Model-theoretic semantics
- Operational semantics
- The frame problem
- Conclusions

LPS = KELPS + Logic Programs

Programs = reactive rules + logic programs with FOL queries.

Computation = model generation.

Model = single, minimal Herbrand model with time stamps.
Truth defined as in classical FOL.

States updated destructively.

LPS framework <***R***, ***L***, ***D***> and current state ***S***

The state ***S*** is a set of ground atomic sentences, representing:

- the extensional part of a deductive database, or
- program variables changed by destructive assignment, or
- a Herbrand model of the current state of the world.

Reactive rules ***R***: $\forall X\ [antecedent(X) \rightarrow \exists Y\ consequent(X, Y)]$

Logic program ***L*** $= L_{int} \cup L_{events} \cup L_{timeless} \cup L_{temp}$

$L_{timeless}$ defines time independent predicates.
L_{int} defines intensional predicates in terms of extensional predicates.
L_{events} defines composite events in terms of atomic events.
L_{temp} defines temporal predicates $<$, $\leq$.

Domain theory ***D*** is a logic program that
defines preconditions and postconditions of atomic events

Blocks world <***R***, ***L***, ***D***> and current state ***S***

R: $request(on(Block, Place), T1) \rightarrow make\text{-}on(Block, Place, T2, T3) \wedge T1 < T2$

S* : Deductive database: extensional predicates *on(Block, Place, T)*.

L_{int}: $clear(table, T)$
$clear(Block, T) \leftarrow \neg \exists X\ on(X, Block, T)$

L_{events}: $make\text{-}on(Block, Place, T, T) \leftarrow on(Block, Place, T)$
$make\text{-}on(Block, Place, T1, T3) \leftarrow make\text{-}clear(Block, TB1, TB2)$
$\wedge\ make\text{-}clear(Place, TP1, TP2) \wedge min(TB1, TP1, T1) \wedge max(TB2, TP2, T2)$
$\wedge\ move(Block, Place, T3) \wedge T2 < T3$

$make\text{-}clear(Place, T, T) \leftarrow clear(Place, T)$
$make\text{-}clear(Place, T1, T3) \leftarrow on(Block, Place, T1)$
$\wedge\ make\text{-}clear(Block, T1, T2) \wedge move(Block, table, T3) \wedge T2 < T3$

D: $possible(move(Block, Place), T) \leftarrow clear(Block, T) \wedge clear(Place, T) \wedge Block \neq Place$
$initiates(move(Block, Place), on(Block, Place), T)$
$terminates(move(Block, Place), on(Block, Support), T) \leftarrow on(Block, Support,T)$

Dialogue/parsing example <***R***, ***L***, ***D***> where ***L*** = L_{events} ***D*** = {} ***S*** = {}

Reactive rule ***R***:

sentence(T1, T2) $\rightarrow$ sentence (T3, T4) $\wedge$ T2 < T3 < T2 + 10

Atomic events:

word(my, 1, 2)   word(name, 2, 3)   word(is, 3, 4)   word(bob, 4, 5)

Composite events (and actions) L_{events} :

adjective(T1, T2) $\leftarrow$ word(my, T1, T2)
adjective(T1, T2) $\leftarrow$ word(your, T1, T2)

noun(T1, T2) $\leftarrow$ word(name, T1, T2)
noun(T1, T2) $\leftarrow$ word(bob, T1, T2)

verb(T1, T2) $\leftarrow$ word(is, T1, T2)
noun(T1, T2) $\leftarrow$ word(what, T1, T2)

sentence(T1, T3) $\leftarrow$ noun-phrase(T1, T2) $\wedge$ verb-phrase(T2, T3)
noun-phrase(T1, T3) $\leftarrow$ adjective(T1, T2) $\wedge$ noun(T2, T3)
noun-phrase(T1, T2) $\leftarrow$ noun(T1, T2)
verb-phrase(T1, T3) $\leftarrow$ verb(T1, T2) $\wedge$ noun-phrase(T2, T3)
verb-phrase(T1, T2) $\leftarrow$ verb(T1, T2)

The reactive rule is true in the sequence of atomic and composite events

{*word(my, 1, 2)*	*word(name, 2, 3)*	*word(is, 3, 4)*
word(bob, 4, 5)	*word(what, 6, 7)*	*word(is, 7, 8)*
word(your, 8, 9)	*word(name, 9, 10)*	
adjective(1, 2)	*noun(2, 3)*	*verb(3, 4)*
noun(4, 5)	*noun(6, 7)*	*verb(7, 8)*
adjective(8, 9)	*noun(9, 10)*	*noun-phrase(1, 3)*
noun-phrase(2, 3)	*noun-phrase(4, 5)*	*verb-phrase(3, 5)*
sentence(2, 4)	*sentence(2, 5)*	*sentence(1, 5)*
noun-phrase(6, 7)	*noun-phrase(8, 10)*	*noun-phrase(9, 10)*
verb-phrase(7, 10)	*sentence(6, 8)*	*sentence(6, 10)*}

$\cup$ ***Temp***

where ***Temp*** (includes 5 < 6) is the extension of the inequality relation defined by L_{temp} .

LPS: alternative external notations

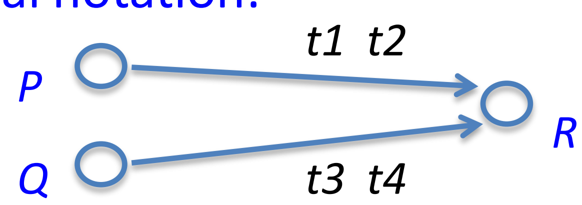
Transaction Logic:

$P \otimes Q$ means $P(T_1) \wedge Q(T_2) \wedge T_1 < T_2$
or $P(T_1, T_2) \wedge Q(T_3, T_4) \wedge T_2 < T_3$

Modal temporal logic:

$P \wedge \Diamond Q$ means $P(T_1) \wedge Q(T_2) \wedge T_1 < T_2$.
$P \wedge \circ Q$ means $P(T) \wedge Q(T+1)$
or $P(T_1) \wedge Q(T_2) \wedge T_1 < T_2 \leq T_1 + \varepsilon$

Graphical notation:



means $P(T_1) \wedge Q(T_2) \wedge R(T_3) \wedge T_1 + t1 \leq T_3 \leq T_1 + t2 \wedge$
$T_2 + t3 \leq T_3 \leq T_2 + t4$

Outline

- KELPS
- Related work
- LPS
- MALPS = Multi Agent LPS (The Dining Philosophers)
- Model-theoretic semantics
- Operational semantics
- The frame problem
- Conclusions

The Dining Philosophers

The Dining Philosophers

The initial state S_0: *available(fork$_0$)*

available(fork$_1$)

available(fork$_2$)

available(fork$_3$)

available(fork$_4$)

$L_{timeless}$ *adjacent(fork$_0$, philosopher(0), fork$_1$)*

adjacent(fork$_1$, philosopher(1), fork$_2$)

adjacent(fork$_2$, philosopher(2), fork$_3$)

adjacent(fork$_3$, philosopher(3), fork$_4$)

adjacent(fork$_4$, philosopher(4), fork$_0$)

The Dining Philosophers – with time-free syntax

time-to-eat(philosopher(I))
$\rightarrow$ *dine(philosopher(I))*

dine(philosopher(I))
$\leftarrow$ *think(philosopher(I)),*
pickup-forks(philosopher(I)),
eat(philosopher(I)),
putdown-forks(philosopher(I))

Atomic actions are defined by the domain specific event theory *D*

pickup-forks(philosopher(I))

terminates $available(F_1)$ and $available(F_2)$

preconditions $available(F_1)$, $available(F_2)$ if
$adjacent(F_1, philosopher(I), F_2)$.

putdown-forks(philosopher(I))

initiates $available(F_1)$ and $available(F_2)$ if
$adjacent(F_1, philosopher(I), F_2)$.

The reactive rule is true in the sequence of states and actions:

S_0: {*available(fork$_0$), available(fork$_1$), available(fork$_2$), available(fork$_3$), available(fork$_4$)*}

A_1: {*think(philosopher(0)), think(philosopher(1)), think(philosopher(2)), think(philosopher(3)), think(philosopher(4))*}

S_1: {*available(fork$_0$), available(fork$_1$), available(fork$_2$), available(fork$_3$), available(fork$_4$)*}

A_2: {*pickup-forks(philosopher(0)), pickup-forks(philosopher(2))*}

S_2: {*available(fork$_4$)*}

A_3: {*eat(philosopher(0)), eat(philosopher(2))*}

S_3: {*available(fork$_4$)*}

A_4: {*putdown-forks(philosopher(0)), putdown-forks(philosopher(2))*}

S_4: {*available(fork$_0$), available(fork$_1$), available(fork$_2$), available(fork$_3$), available(fork$_4$)*}

A_5: {*pickup-forks(philosopher(1)), pickup-forks(philosopher(3))*}

S_5: {*available(fork$_0$)*}

A_6: {*eat(philosopher(1)), eat(philosopher(3))*}

S_6: {*available(fork$_0$)*}

A_7: {*putdown-forks(philosopher(1)), putdown-forks(philosopher(3))*}

S_7: {*available(fork$_0$), available(fork$_1$), available(fork$_2$), available(fork$_3$), available(fork$_4$)*}

A_8: {*pickup-forks(philosopher(4))*}

S_8: {*available(fork$_1$), available(fork$_2$), available(fork$_3$)*}

A_9: {*eat(philosopher(4))*}

S_9: {*available(fork$_1$), available(fork$_2$), available(fork$_3$)*}

A_{10}: {*putdown-forks(philosopher(4))*}

S_{10}: {*available(fork$_0$), available(fork$_1$), available(fork$_2$), available(fork$_3$), available(fork$_4$)*}

Outline

- KELPS
- Related work
- LPS
- MALPS
- Model-theoretic semantics
- Operational semantics
- The frame problem
- Conclusions

Model-theoretic semantics

Given $<\boldsymbol{R}, \boldsymbol{L}, \boldsymbol{D}>$ and initial state $\boldsymbol{S_0}^*$ with explicit time,
$\boldsymbol{ex_1}^*, \ldots, \boldsymbol{ex_i}^*, \ldots$. sequence of sets of external events with explicit time,

the *computational task* is to generate a sequence of sets of actions
$\boldsymbol{a_1}^*, \ldots, \boldsymbol{a_i}^*, \ldots$. such that $\boldsymbol{R}$ is true in the "intended" minimal model of:

$$\boxed{\begin{array}{l} \boldsymbol{L} \cup \boldsymbol{S_0}^* \cup \boldsymbol{S_1}^* \cup \ldots \boldsymbol{S_i}^* \ldots \cup \\ \boldsymbol{e_1}^* \cup \boldsymbol{e_2}^* \cup \ldots \boldsymbol{e_i}^* \cup \ldots \end{array}}$$

where $\boldsymbol{e_i}^* = \boldsymbol{ex_i}^* \cup \boldsymbol{a_i}^*$

$\boldsymbol{S_i} = (\boldsymbol{S_{i-1}} - \{p \mid terminates(\boldsymbol{e_i}, p, \boldsymbol{t_i})$ is true in $\boldsymbol{e_i}^* \cup \boldsymbol{S_{i-1}}^* \cup L_{timeless} \cup L_{int} \cup D\})$
$\cup \{p \mid initiates(\boldsymbol{e_i}, p, \boldsymbol{t_i})$ is true in $\boldsymbol{e_i}^* \cup \boldsymbol{S_{i-1}}^* \cup L_{timeless} \cup L_{int} \cup D\}$

$\boldsymbol{S_i}^* = \{holds(p, \boldsymbol{t_i}) \mid p \in \boldsymbol{S_i}$ at time $\boldsymbol{t_i}\}$

Outline

- KELPS
- Related work
- LPS
- MALPS
- Model-theoretic semantics
- Operational semantics
- The frame problem
- Conclusions

The operational semantics is an observe–decide–think –act cycle.

The *i-th* cycle transforms $\boldsymbol{S_{i\text{-}1}}$, $\boldsymbol{R_{i\text{-}1}}$, $\boldsymbol{G_{i\text{-}1}}$ and $\boldsymbol{e_i}$ into $\boldsymbol{S_i}$, $\boldsymbol{R_i}$, $\boldsymbol{G_i}$, and actions $\boldsymbol{a_{i+1}}$

- $\boldsymbol{G_i}$ is a conjunction of goals.
- Each goal has the form:

$$
\begin{aligned}
& subgoal_{11}(X, Y) \wedge \cdots \wedge subgoal_{1l1}(X, Y) \\
\vee\ & \cdots \\
\vee\ & subgoal_{m1}(X, Y) \wedge \cdots \wedge subgoal_{mlm}(X, Y)
\end{aligned}
$$

Each $subgoal_{ij}$ is an event, FOL condition or temporal constraint.

Simplified operational semantics (for LPS - L_{events})

Step 0. Observe. Use $\boldsymbol{e_i}$ to transform $\boldsymbol{S_{i-1}}$ into $\boldsymbol{S_i}$.

Step 1. Think. If *earlier-antecedents(X)* $\wedge$ *later-antecedents(X)* $\rightarrow$ *consequent(X, Y)*
is in $\boldsymbol{R_{i-1}}$ and *earlier-antecedents(x)* is true in $\boldsymbol{e_i}^* \cup \boldsymbol{S_i}^* \cup L_{timeless} \cup L_{int}$
then simplify any temporal constraints in *later-antecedents(x)* $\rightarrow$ *consequent(x, Y)*
and add the result to $\boldsymbol{R_{i-1}}$ to obtain $\boldsymbol{R_i}$.

If *later-antecedents(x)* is empty, then add the result to $\boldsymbol{G_{i-1}}$ as a new goal.

Step 2.1. Decide. Choose a set ***P*** of plans from one or more goals in $\boldsymbol{G_{i-1}}$.

Step 2.2. Think. For every plan in ***P***, choose a form
earlier-consequents(Y) $\wedge$ *later-consequents(Y).*
If *earlier-consequents(y)* is true in $\boldsymbol{e_i}^* \cup \boldsymbol{S_i}^* \cup L_{timeless} \cup L_{int}$
then simplify any temporal constraints in *later-consequents(y)*
and add the result as an new plan to the same goal in ***P*** to obtain $\boldsymbol{G_i}$.

Step 2.3. Act. For every plan in ***P*** of a form
actions(Z) $\wedge$ *other-consequents(Z),* choose such a form,
attempt to execute *actions(Z)*
and add any successfully executed instances *actions(z)* to $\boldsymbol{e_{i+1}}$.

The operational semantics is sound with respect to the model-theoretic semantics.

Theorem. Given external events $\boldsymbol{ex_1}, \ldots, \boldsymbol{ex_n}, \ldots.,$
suppose the operational semantics generates:

$$S_0, R_0, G_0, a_1, \ldots, S_i, R_i, G_i, a_{i+1}, \ldots.$$

Let $\boldsymbol{M}$ be the "intended" minimal model of:

$$\boxed{\begin{array}{l} L \cup S_0^* \cup S_1^* \cup \ldots. S_i^* \ldots \cup \\ e_1^* \cup e_2^* \cup \ldots e_i^* \cup \ldots. \end{array}}$$

Then $\boldsymbol{R_0} \cup \boldsymbol{G_0}$ is true in $\boldsymbol{M}$ if and only if

for every new goal G added to a goal state $\boldsymbol{G_i}$,
there exists a goal state $\boldsymbol{G_j}$, $j \geq i$ such that the empty plan (equivalent to true) is added as a new plan to the same goal as G in $\boldsymbol{G_j}$.

Incompleteness

The operational semantics is incomplete.
It cannot preventively make a rule true by making its *antecedents* false:

$$attacks(X, me, T_1) \wedge \neg\, prepared\text{-}for\text{-}attack(me, T_1)$$
$$\rightarrow\ surrender(me, T_2) \wedge T_1 < T_2 \leq T_1 + \delta$$

It cannot proactively make a reactive rule true by making its *consequents* true before its *antecedents* become true:

$$enter\text{-}bus(me, T_1)$$
$$\rightarrow have\text{-}ticket(me, T_2) \wedge T_1 < T_2 \leq T_1 + \varepsilon$$

Outline

- KELPS
- Related work
- LPS
- MALPS
- Model-theoretic semantics
- Operational semantics
- The frame problem
- Conclusions

Model-theoretic semantics – an alternative formulation with general purpose event theory $\boldsymbol{Et_{holds}}$

$holds(P, T_2) \leftarrow happens(E, T_1, T_2) \wedge initiates(E, P, T_1)$
$holds(P, T_2) \leftarrow holds(P, T_1) \wedge happens(E, T_1, T_2) \wedge \neg\, terminates(E, P, T_1)$

Given <***R***, ***L***, ***D***> and initial state $\boldsymbol{S_0}^*$ with explicit time,
$\boldsymbol{ex_1}^*, \ldots, \boldsymbol{ex_i}^*, \ldots$. sequence of sets of external events with explicit time,

the *computational task* is to generate a sequence of sets of actions $\boldsymbol{a_1}^*, \ldots, \boldsymbol{a_i}^*, \ldots$. such that ***R*** is true in the "intended" minimal model of:

$$\boxed{\begin{array}{l} \boldsymbol{ET_{holds}} \cup \boldsymbol{L} \cup \boldsymbol{D} \cup \boldsymbol{S_0^*} \cup \\ \boldsymbol{e_1^*} \cup \boldsymbol{e_2^*} \cup \ldots \boldsymbol{e_i^*} \cup \ldots \end{array}}$$

Solving the computational aspect of the frame problem

Theorem. The "intended" minimal model of:

$$\boxed{\begin{array}{l} \boldsymbol{ET_{holds}} \cup \boldsymbol{L} \cup \boldsymbol{D} \cup \boldsymbol{S_0}^* \cup \\ \boldsymbol{e_1}^* \cup \boldsymbol{e_2}^* \cup \ldots \boldsymbol{e_i}^* \cup \ldots. \end{array}}$$

is identical to the "intended" minimal model of:

$$\boxed{\begin{array}{l} \boldsymbol{L} \cup \boldsymbol{D} \cup \boldsymbol{S_0}^* \cup \boldsymbol{S_1}^* \cup \ldots.\ \boldsymbol{S_i}^* \ldots \cup \\ \boldsymbol{e_1}^* \cup \boldsymbol{e_2}^* \cup \ldots \boldsymbol{e_i}^* \cup \ldots. \end{array}}$$

Conclusions

- Destructive assignment does not need a semantics.
  It is the semantics.

- Challenge: Find a framework that unifies

  Programming
  Databases
  AI knowledge representation and problem-solving