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:

\[ \forall X \left( \text{antecedent}(X) \rightarrow \exists Y \text{ consequent}(X, Y) \right) \]

abbreviated

\[ \text{antecedent}(X) \rightarrow \text{consequent}(X, Y) \]

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 $T_1 \land$
smoke detector detects smoke in area A at time $T_2 \land$
$|T_1 - T_2| \leq 60 \text{ sec} \land \max(T_1, T_2, T)$

→ activate local fire suppression in area A at time $T_3 \land T < T_3 \leq T + 10 \text{ sec} \land$
send security guard to area A at time $T_4 \land T_3 < T_4 \leq T_3 + 30 \text{ sec}$

∨ call fire department to area A at time $T_3^{'} \land T < T_3^{'} \leq T + 120 \text{ sec}$
Syntax of reactive rules in KELPS

\[
\text{antecedent}_1(X) \land \cdots \land \text{antecedent}_n(X)
\]

\[
\rightarrow \quad \text{consequent}_{11}(X, Y) \land \cdots \land \text{consequent}_{1l_1}(X, Y)
\]

\[
\lor \quad \cdots \quad \lor \quad \text{consequent}_{m_1}(X, Y) \land \cdots \land \text{consequent}_{m_{l_m}}(X, Y)
\]

Each \(\text{antecedent}_i(X)\) and \(\text{consequent}_{ij}(X, Y)\) 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 \(\text{time}_1 < \text{time}_2\) or \(\text{time}_1 \leq \text{time}_2\)

Each \(\text{consequent}_{i_1}(X, Y) \land \cdots \land \text{consequent}_{i_{l_1}}(X, Y)\) 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:

\[
\text{‘past and present formula’ implies ‘present or future formula’}
\]

**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 \(R\): \(\forall X \ [\text{antecedent}(X) \rightarrow \exists Y \ \text{consequent}(X, Y)]\)

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-on(Block, Place, T2, T3) \(\land\) T1 < T2

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

\(L_{int}:\) clear(table, T)

\(\) clear(Block, T) \(\leftarrow\) \(\exists X\) on(X, Block, T)

\(\) events:

\(\) make-on(Block, Place, T, T) \(\leftarrow\) on(Block, Place, T)

\(\) make-on(Block, Place, T1, T3) \(\leftarrow\) make-clear(Block, TB1, TB2)

\(\) \(\land\) make-clear(Place, TP1, TP2) \(\land\) min(TB1, TP1, T1) \(\land\) max(TB2, TP2, T2)

\(\) \(\land\) move(Block, Place, T3) \(\land\) T2 < T3

\(\) make-clear(Place, T, T) \(\leftarrow\) clear(Place, T)

\(\) make-clear(Place, T1, T3) \(\leftarrow\) on(Block, Place, T1)

\(\) \(\land\) make-clear(Block, T1, T2) \(\land\) move(Block, table, T3) \(\land\) T2 < T3

\(D:\) possible(move(Block, Place), T) \(\leftarrow\) clear(Block, T) \(\land\) clear(Place, T) \(\land\) 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 $L = L_{\text{events}}$ $D = \{\}$ $S = \{\}$

Reactive rule $R$:
sentence(T1, T2) $\rightarrow$ sentence (T3, T4) $\land$ 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) $L_{\text{events}}$:

adjective(T1, T2) $\leftarrow$ word(my, T1, T2)
adjective(T1, T2) $\leftarrow$ word(your, T1, T2)
noun(T1, T2) $\leftarrow$ word(name, T1, T2) verb(T1, T2) $\leftarrow$ word(is, T1, T2)
noun(T1, T2) $\leftarrow$ word(bob, T1, T2) noun(T1, T2) $\leftarrow$ word(what, T1, T2)
sentence(T1, T3) $\leftarrow$ noun-phrase(T1, T2) $\land$ verb-phrase(T2, T3)
noun-phrase(T1, T3) $\leftarrow$ adjective(T1, T2) $\land$ noun(T2, T3)
noun-phrase(T1, T2) $\leftarrow$ noun(T1, T2)
verb-phrase(T1, T3) $\leftarrow$ verb(T1, T2) $\land$ 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

\{ \text{word}(my, 1, 2), \text{word}(bob, 4, 5), \text{word}(your, 8, 9), \text{adjective}(1, 2), \text{noun}(4, 5), \text{adjective}(8, 9), \text{noun-phrase}(2, 3), \text{sentence}(2, 4), \text{noun-phrase}(6, 7), \text{verb-phrase}(7, 10) \\} \cup \text{Temp}

where \text{Temp} (includes 5 < 6) is the extension of the inequality relation defined by \( L_{temp} \).
LPS: alternative external notations

Transaction Logic:

\[ P \otimes Q \text{ means } P(T_1) \land Q(T_2) \land T_1 < T_2 \]

or

\[ P(T_1, T_2) \land Q(T_3, T_4) \land T_2 < T_3 \]

Modal temporal logic:

\[ P \land \Diamond Q \text{ means } P(T_1) \land Q(T_2) \land T_1 < T_2. \]

\[ P \land \Box Q \text{ means } P(T) \land Q(T+1) \]

or

\[ P(T_1) \land Q(T_2) \land T_1 < T_2 \leq T_1 + \varepsilon \]

Graphical notation:

\[ P \]

\[ Q \]

\[ R \]

\[ P \land Q \land R \land T_1 + t_1 \leq T_3 \leq T_1 + t_2 \land T_2 + t_3 \leq T_3 \leq T_2 + t_4 \]
Outline

- KELPS
- Related work
- LPS
- \textbf{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 $S_0$: $\text{available}(\text{fork}_0)$
$\text{available}(\text{fork}_1)$
$\text{available}(\text{fork}_2)$
$\text{available}(\text{fork}_3)$
$\text{available}(\text{fork}_4)$

$L_{\text{timeless}}$
$\text{adjacent}(\text{fork}_0, \text{philosopher}(0), \text{fork}_1)$
$\text{adjacent}(\text{fork}_1, \text{philosopher}(1), \text{fork}_2)$
$\text{adjacent}(\text{fork}_2, \text{philosopher}(2), \text{fork}_3)$
$\text{adjacent}(\text{fork}_3, \text{philosopher}(3), \text{fork}_4)$
$\text{adjacent}(\text{fork}_4, \text{philosopher}(4), \text{fork}_0)$
The Dining Philosophers — with time-free syntax

time-to-eat(philosopher(I))
→ dine(philosopher(I))

dine(philosopher(I))
← 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 $\text{available}(F_1)$ and $\text{available}(F_2)$
- preconditions $\text{available}(F_1)$, $\text{available}(F_2)$ if $\text{adjacent}(F_1, \text{philosopher}(I), F_2)$.

**putdown-forks(philosopher(I))**
- initiates $\text{available}(F_1)$ and $\text{available}(F_2)$ if $\text{adjacent}(F_1, \text{philosopher}(I), F_2)$.
The reactive rule is true in the sequence of states and actions:

\[ S_0: \{ \text{available}(\text{fork}_0), \text{available}(\text{fork}_1), \text{available}(\text{fork}_2), \text{available}(\text{fork}_3), \text{available}(\text{fork}_4) \} \]
\[ A_1: \{ \text{think}(\text{philosopher}(0)), \text{think}(\text{philosopher}(1)), \text{think}(\text{philosopher}(2)), \text{think}(\text{philosopher}(3)), \text{think}(\text{philosopher}(4)) \} \]
\[ S_1: \{ \text{available}(\text{fork}_0), \text{available}(\text{fork}_1), \text{available}(\text{fork}_2), \text{available}(\text{fork}_3), \text{available}(\text{fork}_4) \} \]
\[ A_2: \{ \text{pickup-forks}(\text{philosopher}(0)), \text{pickup-forks}(\text{philosopher}(2)) \} \]
\[ S_2: \{ \text{available}(\text{fork}_4) \} \]
\[ A_3: \{ \text{eat}(\text{philosopher}(0)), \text{eat}(\text{philosopher}(2)) \} \]
\[ S_3: \{ \text{available}(\text{fork}_4) \} \]
\[ A_4: \{ \text{putdown-forks}(\text{philosopher}(0)), \text{putdown-forks}(\text{philosopher}(2)) \} \]
\[ S_4: \{ \text{available}(\text{fork}_0), \text{available}(\text{fork}_1), \text{available}(\text{fork}_2), \text{available}(\text{fork}_3), \text{available}(\text{fork}_4) \} \]
\[ A_5: \{ \text{pickup-forks}(\text{philosopher}(1)), \text{pickup-forks}(\text{philosopher}(3)) \} \]
\[ S_5: \{ \text{available}(\text{fork}_0) \} \]
\[ A_6: \{ \text{eat}(\text{philosopher}(1)), \text{eat}(\text{philosopher}(3)) \} \]
\[ S_6: \{ \text{available}(\text{fork}_0) \} \]
\[ A_7: \{ \text{putdown-forks}(\text{philosopher}(1)), \text{putdown-forks}(\text{philosopher}(3)) \} \]
\[ S_7: \{ \text{available}(\text{fork}_0), \text{available}(\text{fork}_1), \text{available}(\text{fork}_2), \text{available}(\text{fork}_3), \text{available}(\text{fork}_4) \} \]
\[ A_8: \{ \text{pickup-forks}(\text{philosopher}(4)) \} \]
\[ S_8: \{ \text{available}(\text{fork}_1), \text{available}(\text{fork}_2), \text{available}(\text{fork}_3) \} \]
\[ A_9: \{ \text{eat}(\text{philosopher}(4)) \} \]
\[ S_9: \{ \text{available}(\text{fork}_1), \text{available}(\text{fork}_2), \text{available}(\text{fork}_3) \} \]
\[ A_{10}: \{ \text{putdown-forks}(\text{philosopher}(4)) \} \]
\[ S_{10}: \{ \text{available}(\text{fork}_0), \text{available}(\text{fork}_1), \text{available}(\text{fork}_2), \text{available}(\text{fork}_3), \text{available}(\text{fork}_4) \} \]
Outline

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

Given \(<R, L, D>\) and initial state \(S_0^*\) with explicit time, \(ex_1^*, ..., ex_i^*, ....\) sequence of sets of external events with explicit time, the \textit{computational task} is to generate a sequence of sets of actions \(a_1^*, ..., a_i^*, ....\) such that \(R\) is true in the “intended” minimal model of:

\[
L \cup S_0^* \cup S_1^* \cup ... \cup S_i^* ... \cup e_1^* \cup e_2^* \cup ... e_i^* \cup ...
\]

where \(e_i^* = ex_i^* \cup a_i^*\)

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

\(S_i^* = \{ holds(p, t_i) \mid p \in S_i \text{ at time } t_i \}\)
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 \(S_{i-1}, R_{i-1}, G_{i-1}\) and \(e_i\) into \(S_i, R_i, G_i,\) and actions \(a_{i+1}\)

- \(G_i\) is a conjunction of goals.

- Each goal has the form:

\[
\text{subgoal}_{11}(X, Y) \land \cdots \land \text{subgoal}_{1l_1}(X, Y) \\
\lor \cdots \\
\lor \text{subgoal}_{m1}(X, Y) \land \cdots \land \text{subgoal}_{ml_m}(X, Y)
\]

Each \(\text{subgoal}_{ij}\) is an event, FOL condition or temporal constraint.
Simplified operational semantics (for LPS - $L_{events}$)

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

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

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

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

**Step 2.2. Think.** For every plan in $P$, choose a form earlier-consequents($Y$) $\land$ later-consequents($Y$).

If earlier-consequents($y$) is true in $e_i^* \cup S_i^* \cup L_{timeless} \cup L_{int}$ 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 $G_i$.

**Step 2.3. Act.** For every plan in $P$ of a form actions($Z$) $\land$ other-consequents($Z$), choose such a form, attempt to execute actions($Z$) and add any successfully executed instances actions($z$) to $e_{i+1}$.
The operational semantics is sound with respect to the model-theoretic semantics.

**Theorem.** Given external events $\mathbf{ex}_1, ..., \mathbf{ex}_n, ...$, suppose the operational semantics generates:

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

Let $M$ be the “intended” minimal model of:

$$L \cup S_0^* \cup S_1^* \cup ... \cup S_i^* \cup ... \cup e_1^* \cup e_2^* \cup ... e_i^* \cup ...$$

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

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

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

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

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

\[
\text{enter-bus}(\text{me}, T_1) \\
\rightarrow \text{have-ticket}(\text{me}, T_2) \land T_1 < T_2 \leq T_1 + \varepsilon
\]
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 $E_t_{holds}$

$\text{holds}(P, T_2) \leftarrow \text{happens}(E, T_1, T_2) \land \text{initiates}(E, P, T_1)$

$\text{holds}(P, T_2) \leftarrow \text{holds}(P, T_1) \land \text{happens}(E, T_1, T_2) \land \neg \text{terminates}(E, P, T_1)$

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

the computational task is to generate a sequence of sets of actions $a_1^*,..., a_i^*,...$ such that $R$ is true in the “intended” minimal model of:

\[
E_t_{holds} \cup L \cup D \cup S_0^* \cup \bigcup e_1^* \cup \bigcup e_2^* \cup ... \cup \bigcup e_i^* \cup ...\]
Solving the computational aspect of the frame problem

**Theorem.** The “intended” minimal model of:

\[
ET_{\text{holds}} \cup L \cup D \cup S_0^* \cup e_1^* \cup e_2^* \cup ... e_i^* \cup ... \]

is identical to the “intended” minimal model of:

\[
L \cup D \cup S_0^* \cup S_1^* \cup ... S_i^* \cup e_1^* \cup e_2^* \cup ... e_i^* \cup ... \]
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