DOOD: FROM WISHFUL THINKING TO (VIRTUAL) REALITY

Michael Kifer

University at Stony Brook, U.S.A.

DOOD Progress Timeline

1995 — **The 4th DOOD, Singapore**
- *DOOD Conference coming of age*
- *Theoretical base well-developed*
- *Further prototypes with various degrees of DOODness*
- *BULL still talking ...*

1993 — **The 3d DOOD in Phoenix**
- *LDL++, Quixote, F-logic interpreter*
- *BULL talking about a commercial product*

1991 — **The 2nd DOOD in Munich**
- *progress in formalizing both structure and behavior of OO*

1989 — **The 1st DOOD Conference in Kyoto**

1988 — **The idea of DOOD conference conceived**
- *disparate theories*
- *serious doubts about feasibility*

... DOOD Mutants break out from Gov AI Labs...

Around 1975 — **In the beginning ...**
- *Minsky's "Frames"*
- *Declarative vs. Procedural knowledge debates*

DOOD Religions

- **prolog++** (cross between Prolog and C++)
  e.g., LDL++, Coral++
- **message-passing prologs**
  e.g., Quintus Objects, Prolog++ (Logic Programming Associates, UK)
- **logic-based religions**
  The main contenders:
  - Linear logic based approaches
  - Dynamic logic based approaches
  - Rewrite logic based works
  - Altered classical logic
  - New logics (usually upward compatible with classical logic)

  ◇ **`F-logic/Transaction logic suite is the most developed and comprehensive in this category.`**

The Grand Scheme of DOOD

DOOD

Structure

(complex objects, queries, types ISA hierarchy, inheritance)

Behavior

(methods with side-effects, state changes)

F-logic

(Kifer, Lausen, Wu)

Transaction Logic

(Bonner, Kifer)

F-logic: Complex Structure the Easy Way

Object description:

$$john[name \rightarrow \text{“}JohnDoe\text{”}; spouse \rightarrow mary; children \twoheadrightarrow \{bob, alice\}]$$
$$mary[name \rightarrow \text{“}MaryDoe\text{”}; spouse \rightarrow john; age \rightarrow 30; children \twoheadrightarrow \{bob, alice\}]$$

ISA hierarchy:

$john : student$ ($john$ is a $student$)
$mary : person$
$student :: person$ ($student$ is a subclass of $person$)

- Can combine the above using $\wedge$, $\vee$, $\neg$, etc.
  In particular, attributes and even class hierarchies can be defined via logical rules:

$$nil : list(T)$$
$$< X|L >: list(T) \leftarrow X : T \wedge L : list(T)$$
$$list(T) :: list(S) \leftarrow T :: S$$

F-logic (contd.)

Methods ($i.e.$, functions that take objects as arguments)

$$P[ageAsOf@Y \rightarrow Y - B] \leftarrow P : person \wedge P[yearOfBirth \rightarrow B]$$
$$L[append@nil \rightarrow L] \leftarrow L : list(T)$$
$$< X|L > [append@M \rightarrow < X|N >] \leftarrow L[append@M \rightarrow N] \wedge X : T \wedge L : list(T)$$

Queries

$$?- \ john[ageAsOf@1989 \rightarrow Y; children \twoheadrightarrow C] \wedge C[yearOfBirth \rightarrow B] \wedge B > 1980$$
$$?- \ < 1, 2, 3 > [append@ < 5, 6, 7 > \rightarrow A]$$

- F-logic methods can only query — no updates
(which is why Transaction Logic is here!)

F-logic (contd.)

Typing (actually, type signatures):

$$
\begin{array}{ll}
person[& ageAsOf \Rightarrow int; \\
 & name \Rightarrow string; \\
 & yearOfBirth \Rightarrow int; \\
 & spouse \Rightarrow person; \\
 & children \Rightarrow\!\!\!\!\!\Rightarrow person]
\end{array}
$$

$$
list(T)[\ append@list(T) \Rightarrow list(T)]
$$

- Can define signatures via deductive rules, query the type structures, etc.
- Type correctness has formal meaning

F-logic's Bullet List

- Logical semantics (General and Herbrand)
- Proof theory
- Upward compatible with classical logic
- Complex objects
- Semantics for type-correctness
- Semantics for structural and behavioral inheritance
- Semantics for encapsulation

♠ Details: JACM, July 1995

Transaction Logic: A Conquest of Change

Why new logic?

- State changes are notoriously hard to capture in logic.
- Reasoning about change vs. specifying change and executing it.
- Harmful (for O-O) distinction between actions and queries.
- Inefficient elementary updates.
- No subroutines!

♠ Transaction logic overcomes all of these and more in a simple, uniform way.

Transaction Logic's Basic Ideas

- **Syntax:**
  - serial conjunction, $\otimes$:
    $\boldsymbol{a} \otimes \boldsymbol{b}$ - do a then do b (plus the "usual" $\wedge$, $\vee$, $\neg$, $\forall$, $\ldots$)
- **Semantics:**
  - Transaction execution paths (sequences of database states)
  - Elementary state transitions.
  - Truth on a path $\equiv$ execution along the path.
- **Proof theory:**
  - proves and executes actions.

Example

- stack a pyramid of N blocks on top of block X:

$$
\begin{aligned}
&stack(0, X) \\
&stack(N, X) \leftarrow N > 0 \otimes move(Y, X) \otimes stack(N-1, Y)
\end{aligned}
$$

$$
\begin{aligned}
&move(X, Y) \leftarrow pickup(X) \otimes putdown(X, Y) \\
&pickup(X) \leftarrow clear(X) \otimes on(X, Y) \otimes on.del(X, Y) \otimes clear.ins(Y) \\
&putdown(X, Y) \leftarrow wider(Y, X) \otimes clear(Y) \otimes on.ins(X, Y) \otimes clear.del(Y)
\end{aligned}
$$

- **Query:** Stack a pyramid of 20 blocks.

$$?- \; stack(20, blkC)$$

Transaction Logic's House Specialties

- Active rules (what did *you* think?)
- Hypothetical reasoning (two new modal operators)
- Subjunctive and counterfactual queries
- Bulk updates
- Updates to arbitrary (not only relational) states
- Allen-style temporal constraints
- Planning
- Avoids the frame problem (for execution and planning)

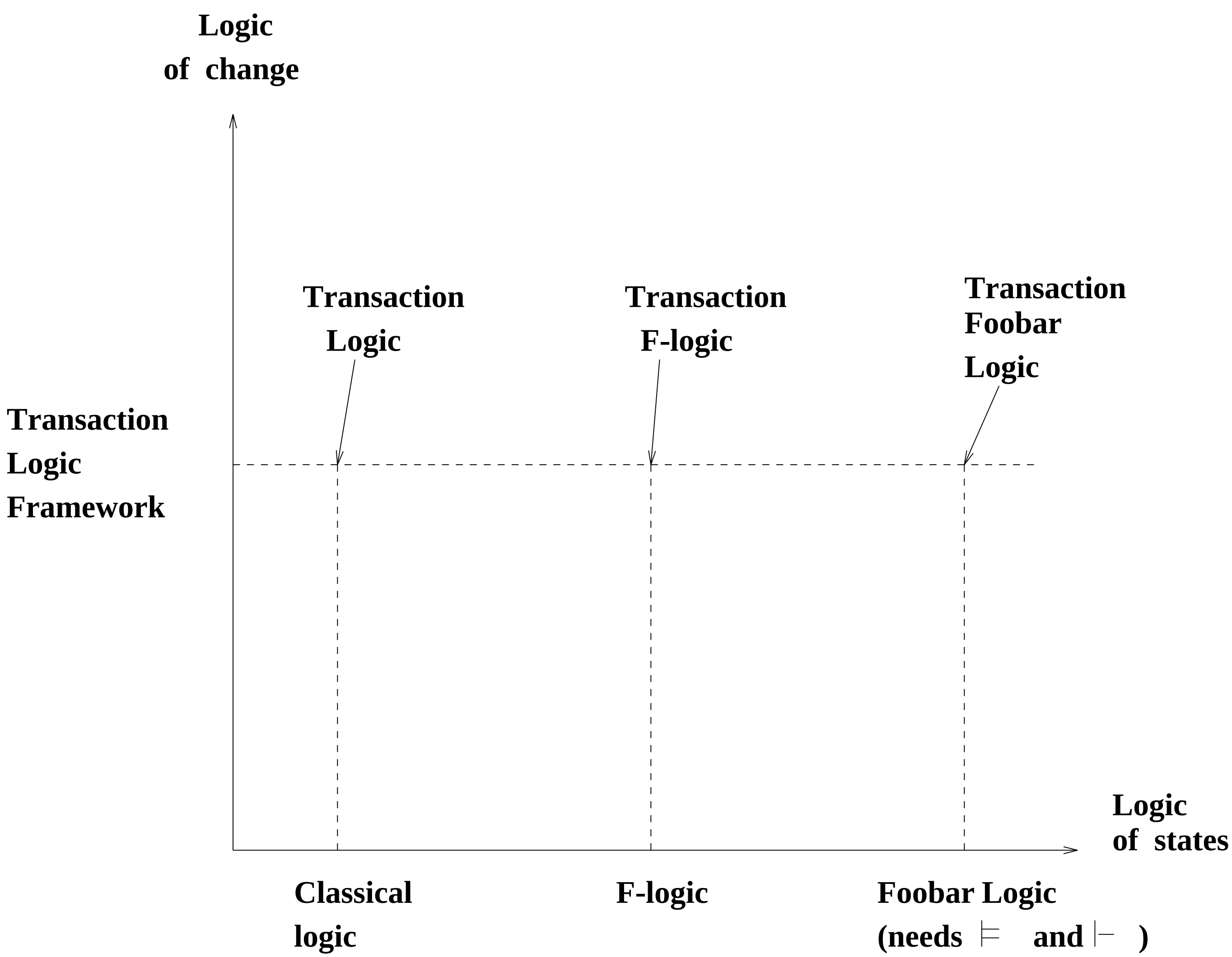
Transaction F-logic: Easier than you may have thought

□ Basic ideas:

- serial conjunction applies to object formulas
- elementary state transitions can be caused by executing methods

♠ Methods can query, change state, or both (*i.e.*, they can be queries with side effect).

Easier than you may have thought (contd.)



Transaction Logic as a Framework

Example

- Stacking pyramids object-oriented way:

$$
\begin{aligned}
&R[stack@N, BaseBlk \rightarrow nil] \leftarrow N > 0 \otimes R[move@Frm, BaseBlk \rightarrow TmpBlk \\
&\qquad\qquad\qquad\qquad\qquad\qquad \otimes\ stack@N-1, TmpBlk \rightarrow nil] \\
&R[move@Frm, To \rightarrow Blk] \leftarrow R[pickup@Blk \rightarrow Frm \otimes putdown@Blk \rightarrow To] \\
&R[pickup@Blk \rightarrow Frm] \leftarrow R : robot \otimes Blk : block \otimes Frm : block \\
&\qquad \otimes Blk[top \rightarrow clear; bottom \rightarrow Frm] \otimes R[state \rightarrow idle] \\
&\qquad \otimes Blk[bottom.del \rightarrow Frm] \otimes Frm[top.ins \rightarrow clear] \\
&\qquad \otimes R[state.replace \rightarrow holding] \\
&R[putdown@Blk \rightarrow To] \leftarrow R : robot \otimes Blk : block \otimes To : block \otimes To[top \rightarrow clear] \\
&\qquad \otimes wider(To, Blk) \otimes R[state \rightarrow holding] \\
&\qquad \otimes Blk[bottom.ins \rightarrow To] \otimes To[top.del \rightarrow clear] \\
&\qquad \otimes R[state.replace \rightarrow idle]
\end{aligned}
$$

♠ This program is longer than the one before because it does a bit more.

Conclusion

DOOD is within reach:

- Theory is there
- Early prototypes are out:
  - Mike Lawley's F-logic interpreter.
  - Georg Lausen's FL System (F-logic).
  - Dave Warren's XSB (HiLog and C-logic, a subset of F-logic).
  - Tony Bonner's Transaction Logic interpreter.

▷ All these are reachable via **`http://www.cs.sunysb.edu/~kifer/dood/`**

... and is exciting:

- Vast field for research (all those problems that I am afraid to touch :-)