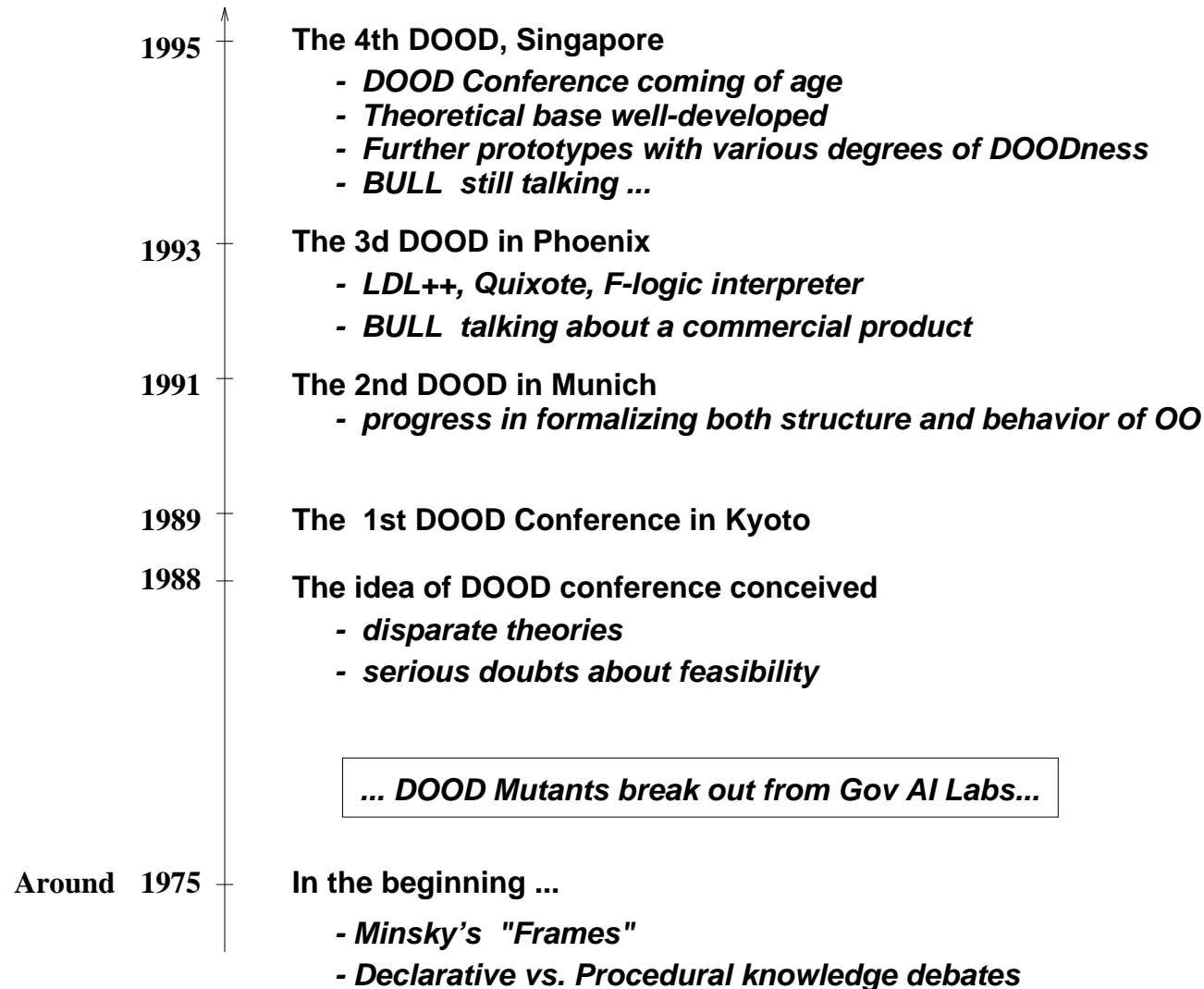


DOOD:
FROM WISHFUL THINKING
TO (VIRTUAL) REALITY

Michael Kifer

University at Stony Brook, U.S.A.

DOOD Progress Timeline



DOOD Religions

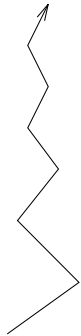
- **prolog++** (cross between Prolog and C++)
e.g., LDL++, Coral++
- **message-passing prologs**
e.g., Quintus Objects, Prolog++ (Logic Programming Associates, UK)
- **logic-based religions**
The main contenders:
 - Linear logic based approaches
 - Dynamic logic based approaches
 - Rewrite logic based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

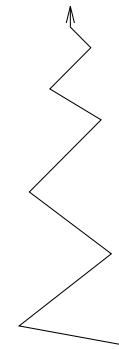


F-logic

(Kifer, Lausen, Wu)

Behavior

***(methods with side-effects,
state changes)***



Transaction Logic

(Bonner, Kifer)

F-logic: Complex Structure the Easy Way

Object description:

$john[name \rightarrow "JohnDoe"; spouse \rightarrow mary; children \rightarrow \{bob, alice\}]$

$mary[name \rightarrow "MaryDoe"; spouse \rightarrow john; age \rightarrow 30; children \rightarrow \{bob, alice\}]$

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:

$nil : list(T)$

$\langle X|L \rangle : list(T) \leftarrow X : T \wedge L : list(T)$

$list(T) :: list(S) \leftarrow T :: S$

F-logic (contd.)

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

$$P[\textit{ageAsOf}@Y \rightarrow Y - B] \leftarrow P : \textit{person} \wedge P[\textit{yearOfBirth} \rightarrow B]$$

$$L[\textit{append}@nil \rightarrow L] \leftarrow L : \textit{list}(T)$$

$$\langle X|L \rangle [\textit{append}@M \rightarrow \langle X|N \rangle] \leftarrow L[\textit{append}@M \rightarrow N] \wedge X : T \wedge L : \textit{list}(T)$$

Queries

$$?- \textit{john}[\textit{ageAsOf}@1989 \rightarrow Y; \textit{children} \rightarrow C] \wedge C[\textit{yearOfBirth} \rightarrow B] \wedge B > 1980$$

$$?- \langle 1, 2, 3 \rangle [\textit{append}@ \langle 5, 6, 7 \rangle \rightarrow A]$$

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

F-logic (contd.)

Typing (actually, type signatures):

```
person[ ageAsOf⇒int;  
         name⇒string;  
         yearOfBirth⇒int;  
         spouse⇒person;  
         children⇒⇒person]
```

```
list(T)[ append@list(T)⇒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 :

- $a \otimes b$ - do a then do b (plus the “usual” $\wedge, \vee, \neg, \forall, \dots$)

- **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 :

$$stack(0, X)$$

$$stack(N, X) \leftarrow N > 0 \otimes move(Y, X) \otimes stack(N - 1, Y)$$

$$move(X, Y) \leftarrow pickup(X) \otimes putdown(X, Y)$$

$$pickup(X) \leftarrow clear(X) \otimes on(X, Y) \otimes on.del(X, Y) \otimes clear.ins(Y)$$

$$putdown(X, Y) \leftarrow wider(Y, X) \otimes clear(Y) \otimes on.ins(X, Y) \otimes clear.del(Y)$$

- **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)

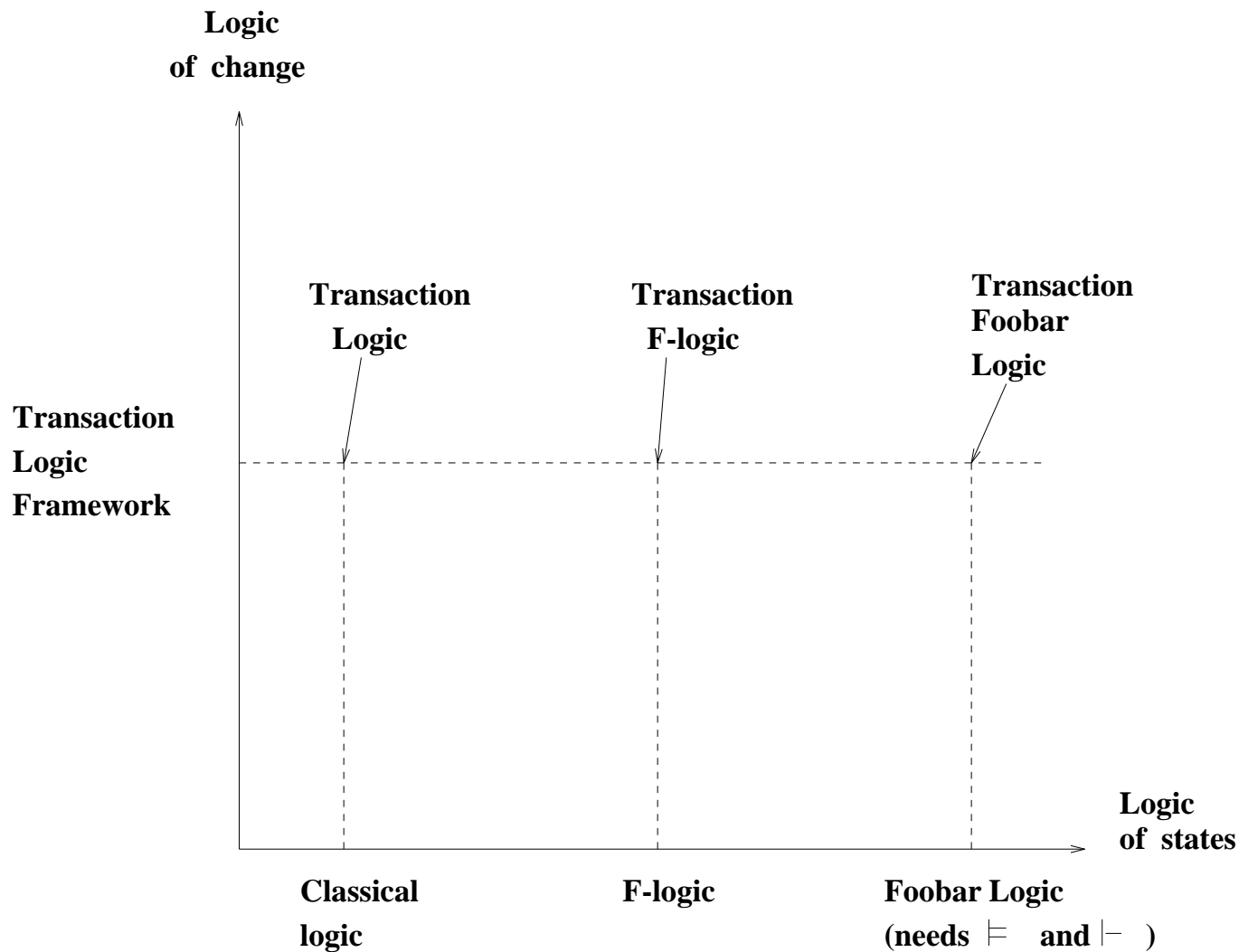
Transaction F-logic: Easier than you may have thought

□ 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:

$$R[\textit{stack}@N, \textit{BaseBlk} \rightarrow \textit{nil}] \leftarrow N > 0 \otimes R[\textit{move}@Frm, \textit{BaseBlk} \rightarrow \textit{TmpBlk} \\ \otimes \textit{stack}@N - 1, \textit{TmpBlk} \rightarrow \textit{nil}]$$

$$R[\textit{move}@Frm, \textit{To} \rightarrow \textit{Blk}] \leftarrow R[\textit{pickup}@Blk \rightarrow Frm \otimes \textit{putdown}@Blk \rightarrow To]$$

$$R[\textit{pickup}@Blk \rightarrow Frm] \leftarrow R : \textit{robot} \otimes \textit{Blk} : \textit{block} \otimes \textit{Frm} : \textit{block} \\ \otimes \textit{Blk}[\textit{top} \rightarrow \textit{clear}; \textit{bottom} \rightarrow \textit{Frm}] \otimes R[\textit{state} \rightarrow \textit{idle}] \\ \otimes \textit{Blk}[\textit{bottom.del} \rightarrow \textit{Frm}] \otimes \textit{Frm}[\textit{top.ins} \rightarrow \textit{clear}] \\ \otimes R[\textit{state.replace} \rightarrow \textit{holding}]$$

$$R[\textit{putdown}@Blk \rightarrow To] \leftarrow R : \textit{robot} \otimes \textit{Blk} : \textit{block} \otimes \textit{To} : \textit{block} \otimes \textit{To}[\textit{top} \rightarrow \textit{clear}] \\ \otimes \textit{wider}(\textit{To}, \textit{Blk}) \otimes R[\textit{state} \rightarrow \textit{holding}] \\ \otimes \textit{Blk}[\textit{bottom.ins} \rightarrow \textit{To}] \otimes \textit{To}[\textit{top.del} \rightarrow \textit{clear}] \\ \otimes R[\textit{state.replace} \rightarrow \textit{idle}]$$

- ♠ 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 :-)