

# LAIR

L. Michelle   P. Roos   P. Shakarian   G. Stoker

Department of Computer Science  
Universities of Maryland

CMSC 818g - Spring 2009

# LAIR Semester Goals

- 1 Implementation of a **context collector** that can successfully extract personal and social network info from Facebook to a SQL data base for representation.
- 2 Implementation of a **compelling Facebook application** used during context collection.
- 3 Establishment of a **first-order logic-based formalism** for reasoning about context over time, to include social networking information.
- 4 Implementation of a **programming interface** that can view SQL data in our formalism described above.
- 5 Demonstrate **information visualization techniques** on a contextual domain based on our formalism that allow the user to derive context.

### CONTEXT

Affiliation Network + Global Propositions + Reasoning Logic



*All Over Time*

- Moved to a non-probabilistic model based on a **Kripke structure** (state machine) extracted from a database as a structure and **LTL** formulae as specifications

# Formalism

## Reasoning

**Linear Temporal Logic** (LTL) is a modal temporal logic with modalities referring to time.

Consists of:

- predicates
- logic connectives  $\neg, \wedge, \vee, \rightarrow$
- temporal modal operators: eventually, next, always, ...

In LTL, one can encode formulae about the future of paths such as that a condition will eventually be true, that a condition will be true until another fact becomes true, etc.

## LTL for Context

Time is a critical component of context. Other approaches that utilize first-order logic to describe context often use time in an ad-hoc manner. By the use of LTL, however, we can provide a more rigorous time component to our descriptions of context. Graph structures inevitably will become integrated into context for context implementation in many applications.

There are two major contributions to this framework.

- 1 Provide a LTL-based framework for context
- 2 Allow description of and reasoning about graphs that change over time incorporating other variables in the environment that may relate to the graph

# Formalism

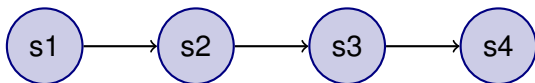
## Structure

A **Kripkie structure** is simply a non-probabilistic finite state transition system. In our use, it will refer to a single time-sequence of states. Hence we will use the tuple  $K = (t_m, A, S)$  where:

- $t_m$  is the duration of time. This structure represents a sequence of events over the time period  $0 \dots t_m$
- $A$  is the set of atoms
- $S$  is the set of states. We define it here simply as a mapping  $S : [0, \dots, t_m] \rightarrow 2^A$ . Hence,  $S(t)$  refers to the atoms true at time  $t$ .

# Formalism

## States



Each state will essentially be a graph structure/affiliation network comprised by atoms, to be used by LTL:

- **Relationship Predicate:** For node space  $N$ , and for  $a, b \in N$  where  $a \neq b$ ,  $rp(a, b, v)$  is true if there is a relationship between  $a$  and  $b$ .
- **Global Predicate:** The set of *global atomic propositions* - atoms that can be true for all users (i.e. universal time)
- **Node Predicate:** The set of atomic propositions specific to a node.

# Assumptions

We aim to create a simple framework for describing context over time, to include changes in graph structure. This framework will have the following characteristics:

- 1 The framework will be non-probabilistic. Hence, a future state of a graph is either possible, or not.
- 2 The structure, to be extracted, will be a Kripkie structure.
- 3 The reasoning framework will be based on LTL, there is a wide variety of LTL software available.
- 4 The graphs in question will not be directed.
- 5 The time period for which a social relationship is valid is determined in the implementation. For example, if relationship  $a$  existed at time  $t - 1$  and the time period is greater than 1, relationship  $a$  is still included in a state at time  $t$ . This is not formalized in the logic.

# Reality Mining Data Set

- Used MIT's Reality cell-phone data set
- <http://reality.media.mit.edu/>
- 97 students, faculty, and researchers over 2004-2005 academic year
- Durations and times of phone calls between pairs of subjects
- Missed Calls, Text Messages
- Time in which pairs of subjects were in proximity of one another (bluetooth)

# Implementation

- Data stored in MySQL, Kripkie structure extracted using program written in Python
- Python program also extracts an initialization file, mapping atoms to relationships in the node space. In this way, we avoid double exponential blow-up caused by node space as we only extract the relationships that exist.

## Implementation

- LairQ Java method takes the initialization file and translates queries in LTL for use with the Kripkie structure (i.e. the user enters LTL formulae with relationships, and the LairQ returns an LTL formula with atoms. If a relationship does not exist, LairQ replaces that atom with FALSE.
- LairQ can also extract special queries designed by the team (next slide).
- LairQ put the new formulae in the same SMV file as the Kripkie structure. You open the file in NuSMV and it tells you if the formula are TRUE.

# Atoms

- dur: Assigns an weight to each edge w. the time for a phone call. i.e.  $\text{dur}[5,24] > 10$  means that there was call between node 5 and 24 greater than 10 in the current state.
- prox: Returns true if two nodes are physically close together. i.e.  $\text{prox}[5,24]$  means that 5 is close to 24 in the current state.
- textmsg: Returns true if a text message was transmitted between two nodes in the current state.
- missedcall: Returns true if there was a missed call between two nodes in the current state.
- weekend: Returns true if it is a weekend day in the current state.

## Queries

- Any LTL, CTL, PSL, formula available in NuSMV
- **Must-happen( $f,g$ )** Any time  $f$  occurs, then  $g$  must follow sometime in the future.
- **Possibly-happen( $f,g$ )** At least one time when  $f$  occurs, then  $g$  must follow sometime in the future.
- **Must-happen-on-weekend( $f$ )** On one of the days each weekend,  $f$  must occur.
- **Possibly-happen-on-weekend( $f$ )** For at least one weekend day,  $f$  occurs.

## Outstanding Issues

- Export to VisuaLink - user can generate more complex queries based on hypotheses from the visualization. The user would then test the hypotheses by querying the Kripkie structure with LTL.
- Implementation of advanced queries
- Completion of Context Collector
- Completion of Inrcircle
- Future work would include building the Kripkie structure from Facebook data and/or automatically generating query rules that could improve performance and usability of a system.

# Demo