Synthesis of Moore Machines from LTL Specifications

Sakari Ellonen

Aalto University

12th November 2010

Outline

1. **Introduction**
2. **The Synthesis Problem**
   - Key Concepts
   - Problem Statement
3. **Synthesis Algorithms**
   - Automata on Infinite Words
   - Two-player Games
   - Satisfiability Algorithm
   - Results

Introduction

Motivation

- Reliability of computer systems is increasingly important
- Complexity renders simulation and testing insufficient for debugging

Formal methods

Methods characterized by

- precise and mathematically rigorous language
- sound rules of inference

Improved debugging and thus improved confidence in designs.

Automated Synthesis

Abstract view

- Input: Specification in LTL
- Output: Reactive system design as a Moore machine

Pros

- Programming by hand made redundant?
- Analysis of specification via a model

Cons

- Discouraging complexity results
- Lack of implementation

LTL Specifications

Linear Temporal Logic

- Widely used, especially in model checking
- Intuitive and simple, yet expressive
- e.g. G(request $\rightarrow$ F(grant))
- Somewhat too abstract for complete specification (e.g. ordering)

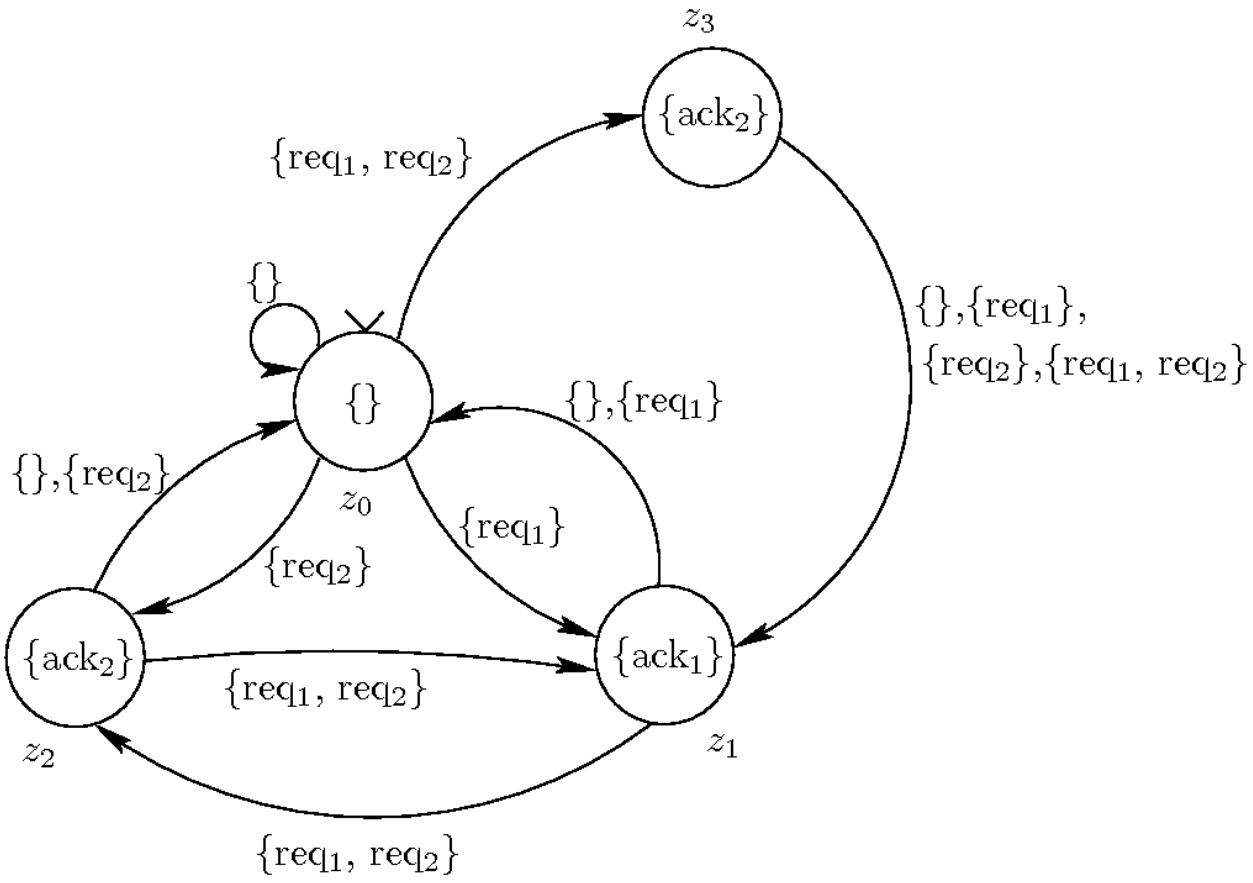
Moore machines

Definition

Finite state machine with output.

- Moore machine $M = (I, O, S, s^0, \eta, \lambda)$, where
- I is the input alphabet, O the output alphabet
- S is the set of states, s^0 the initial state
- $\eta : S \times I \mapsto S$ is the transition function
- $\lambda : S \mapsto O$ is the output function

Moore machine example



$I = \{\{\}, \{req_1\}, \{req_2\}, \{req_1, req_2\}\}$

$O = \{\{\}, \{ack_1\}, \{ack_2\}, \{ack_1, ack_2\}\}$

The Synthesis Problem

Formal definition

Given an LTL formula φ over input alphabet I and output alphabet O, decide whether there exists a Moore machine M such that all words in the language of M satisfy φ.

Automata on Infinite Words

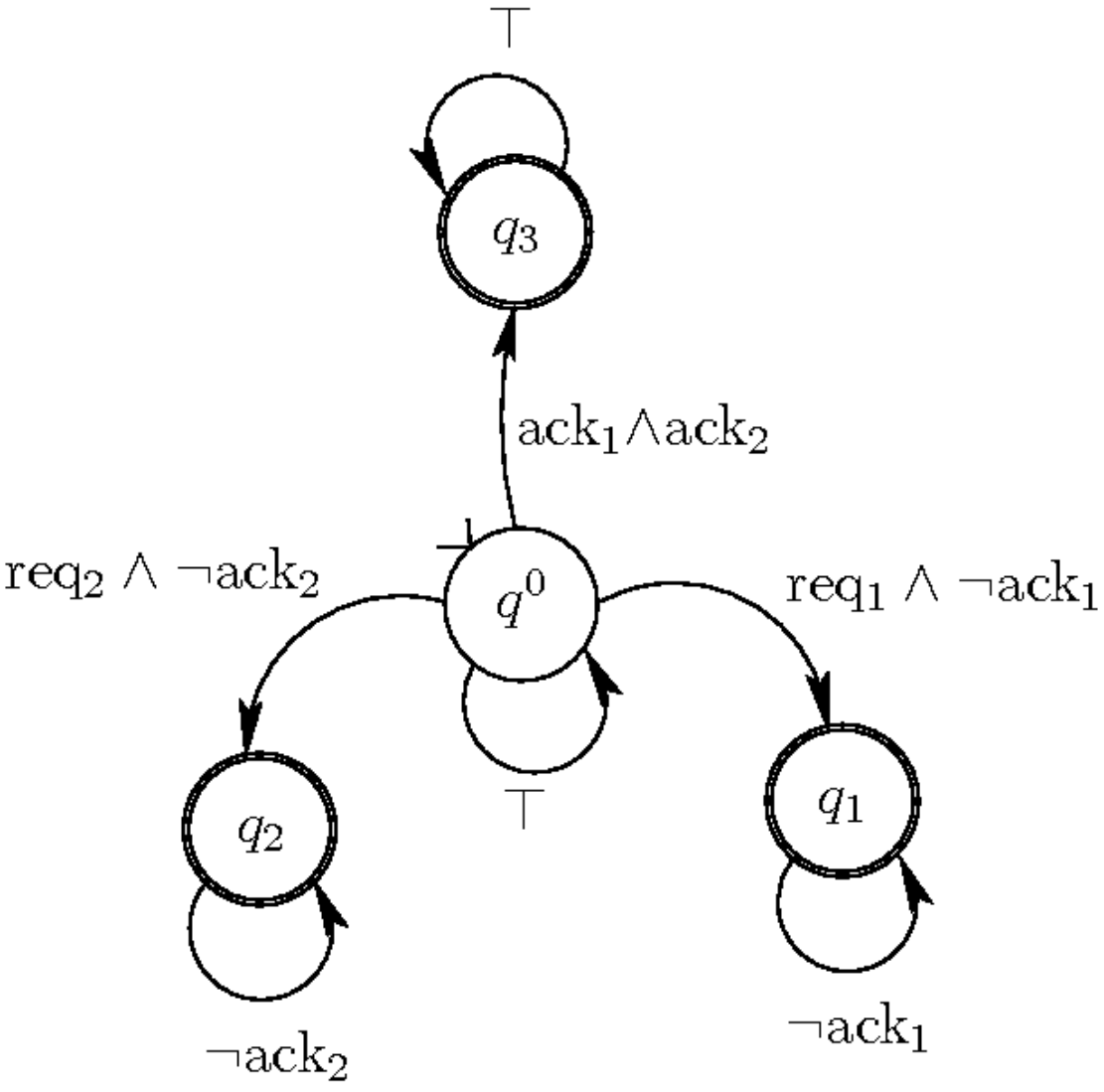
Büchi Automaton Definition

- Tuple $A = (\Sigma, Q, q^0, \delta, \alpha)$ such that
- Σ is the alphabet
- Q the set of states
- $q^0 \in Q$ the initial state
- $\delta \subseteq Q \times \Sigma \times Q$ the transition relation
- $\alpha \subseteq Q$ is the acceptance condition.

An automaton takes a word as input and either accepts or rejects it.

- A run ρ on word $w = w_0, w_1, \ldots$ is a sequence of states $q_0, q_1, \ldots$ such that $\forall i > 0, (q_{i-1}, w_{i-1}, q_i) \in \delta$.
- A word w is accepted iff there is a run ρ on it such that ρ contains infinitely many states in α.

Büchi automaton illustration



Relevance to Synthesis

Automata in Synthesis

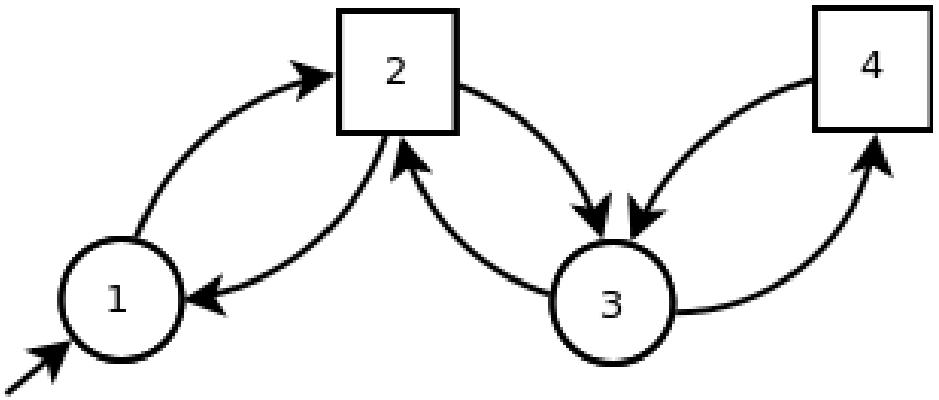
- The specifications we are interested in can be expressed as automata
- All our synthesis algorithms manipulate automata (in fact their negations) instead of LTL formulae.
- Lots of translation tools available, some of which even work
- We are using `spot`

Two-player Games

Definition

Games are directed graphs where

- the set of nodes is partitioned into two (e.g. circles and boxes).
- A token is initially placed on an initial vertex.
- The player who owns the current node decides the successor



Two-player Games continued

Definition continued

- A *play* is an infinite sequence of nodes
- A *winning condition* determines who wins a play
- A memoryless *strategy* associates a move to each node for one of the players
- A strategy is winning if the player always wins by following it
- Solving a game is to determine which one of the players has a winning strategy

Two-player Games continued

Two winning conditions

- *Parity* condition: each node is given a priority, and players called Even and Odd. If the highest priority occurring infinitely often is even, player Even wins, otherwise Odd wins.
- *Safety* condition: a subset of nodes is regarded unsafe. If the play never touches an unsafe state, the protagonist wins, otherwise the antagonist wins.

Two-player Games continued

Relevance to Synthesis

- The Synthesis problem may be reduced to solving a game
- The problem instance (specification) defines the game graph

We need algorithms to

1. map a specification into a game graph
2. solve the game

In our methods the step 1. is considerably more difficult.

Algorithms

Determinization Algorithm

A well-known determinization algorithm for automata results in a parity automaton, which is trivially translated into a game.

Safety Game Algorithm

- The subset construction augmented with acceptance *counters*
- For each state in a subset, the counter keeps track how many acceptance states a run may have seen
- If the environment is able to force k visits to the acceptance sets, he wins
- Should we find a strategy that allows any play to visit the acceptance set at most $k - 1$ times, that strategy is an implementation

Integer Difference Logic method

Reduction to Integer Difference Logic (IDL)

- IDL extends propositional logic by integer variables and constraints of the form $x - y < k$, where x, y are variables and k constant.
- An LTL specification can be mapped into an IDL formula that is satisfiable iff the specification is synthesizable.
- Advantage: produces minimal Moore machines.

Experiments

Table: Greatest n for which the formula was solved by the algorithms

formula	Game 1 (classical)	IDL	Game 2 (ours)	Acacia	Lily
$\varphi^n_{\text{arbiter}}$	2	3	6	4	5
$\varphi^n_{\text{G F}}$	2	2	4	4	4
φ^n_{lift}	-	-	5	3	4

Conclusion

Conclusion

- Automated synthesis has recently regained considerable research interest (eight papers in CAV'10).
- Our algorithm rather competetive, but does not scale up anyway.
- Personal view: we should develop a specification language with synthesis in mind.