Synthesizing Statecharts from Live Sequence Charts

Hillel Kugler

Weizmann Institute of Science

Joint work with David Harel and Amir Pnueli

Omega Meeting, Paris, 3-5 March 2004

The General Approach

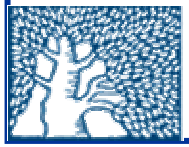
- User specifies scenario-based requirements using Live Sequence Charts
- Tool checks LSC Consistency, provides feedback if inconsistent
- Tool synthesizes a Statechart based model
- Existing tools can be used to generate code and execute model
- Verification methods (Omega) can be used to verify correctness

The Main Challenges

- Get detailed requirements to allow synthesis of interesting and useful models
- Make sure users understand the scenarios and express what they wanted
- Computational complexity of synthesis algorithms
- The synthesized model should be understandable by user
- Support modification of requirements and integration with existing code

Talk Outline

- LSCs (very brief overview)
- Consistency Checking
- Synthesis approach
- Statechart Synthesis (+ DEMO)

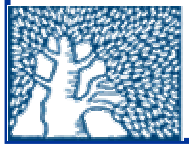


Live sequence charts (LSC's)

"LSC's: Breathing Life into
Message Sequence Charts"

(Damm & Harel, '98)

A natural extension of classical MSCs, with **modalities** (universal/existential, hot/cold, etc.) and **structure** (subcharts, conditionals, loops, etc.)



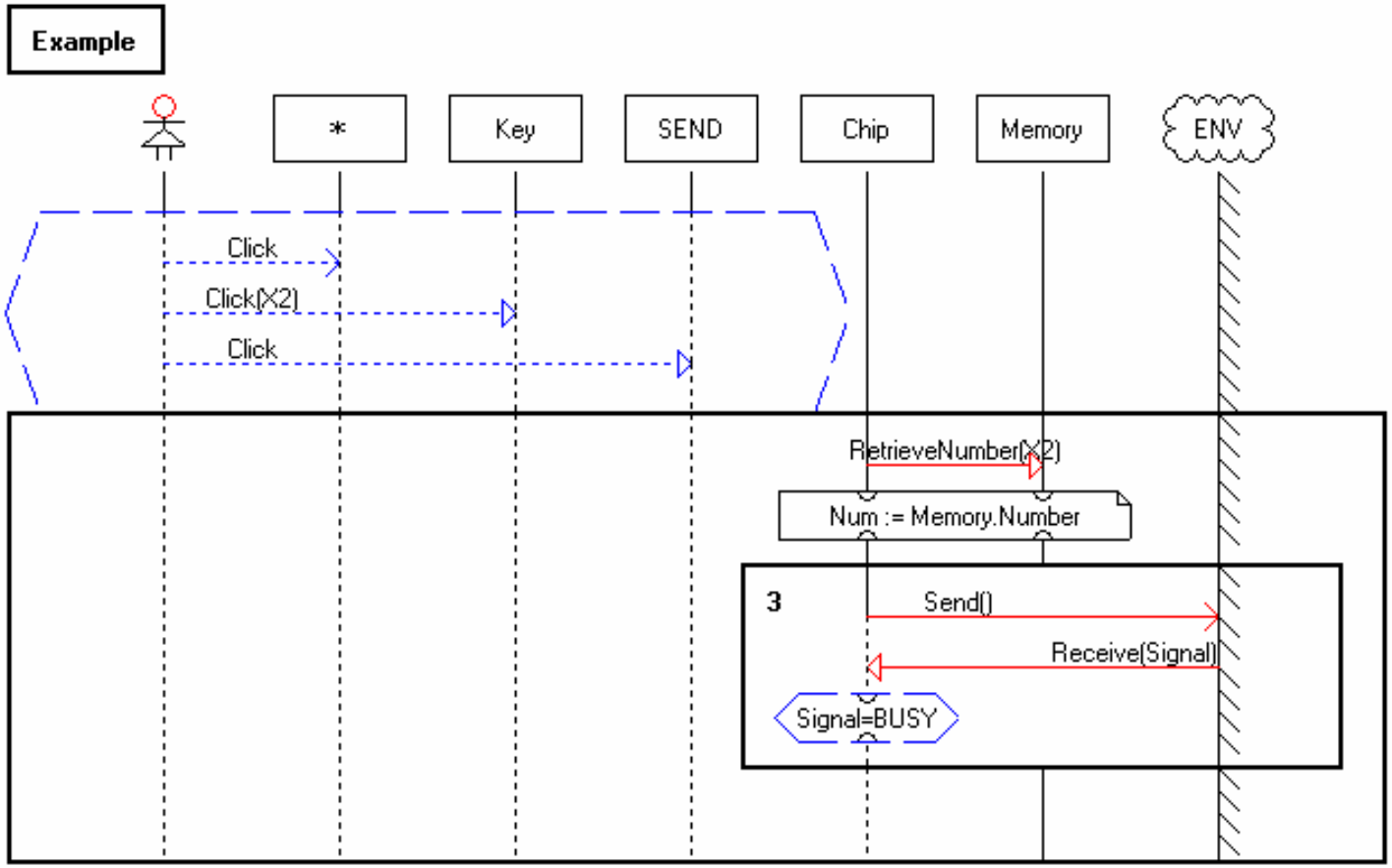
element	mandatory	provisional
chart	universal ∀ all system runs satisfy chart	existential ∃ at least one run satisfies chart

element	mandatory	provisional
chart	universal ∀ all system runs satisfy chart	existential ∃ at least one run satisfies chart
location in chart	hot run of instance must progress beyond location	cold run of instance need not progress beyond location
message	hot if message is sent it must be received	cold receipt of message is not guaranteed
condition	hot condition must be met otherwise abort	cold if condition not met exit current (sub) chart

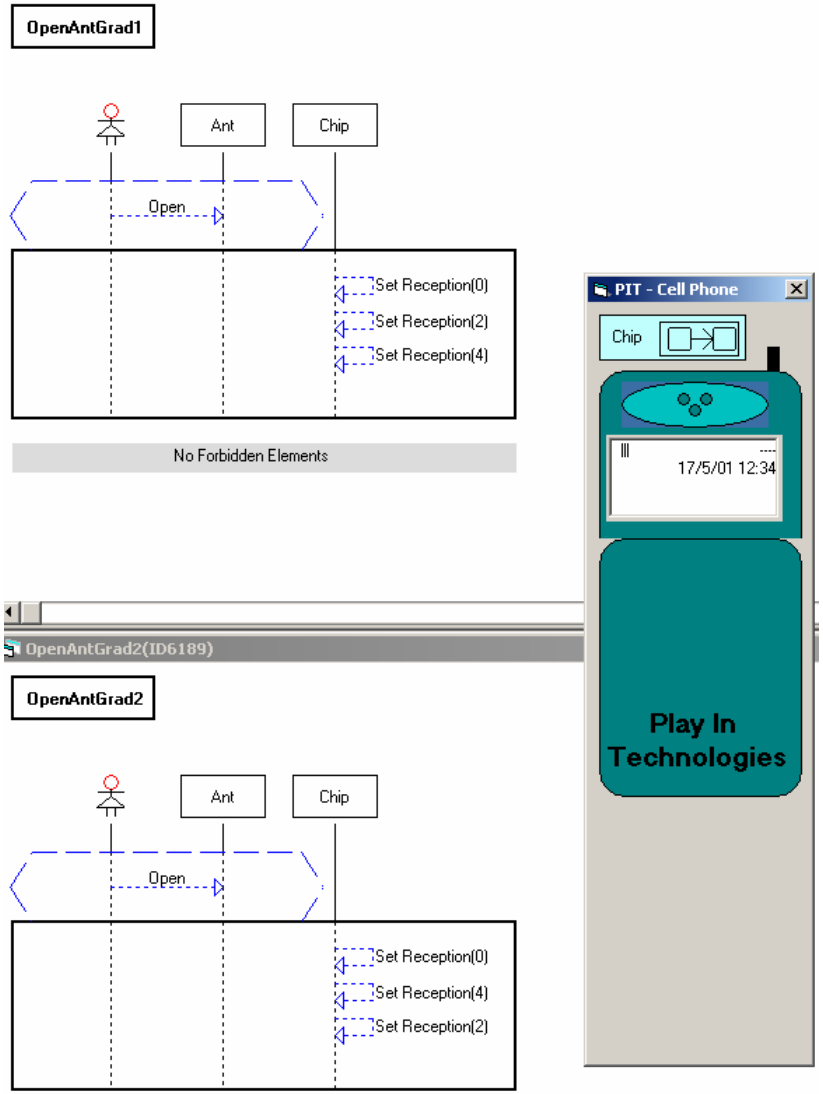
Basic form of an LSC

prechart (if)

main chart (then)



Consistency of LSCs



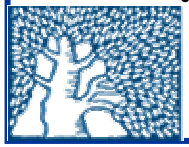
Inconsistent LSCs

- Example shown – two contradicting charts
- General case – many different charts :
  - A chart does not progress beyond hot location
  - Hot condition violated
  - Forbidden element occurs
  - Anti scenario occurs
- Consistency – Game View

Synthesis of Reactive Systems

- Game between two players:
  - Environment (User, Env, External Object)
  - System (All other objects)
- Game :

  Repeat Forever
  - Environment makes a move
  - System performs a superstep
- Consistency – exists a strategy for the system such that LSC spec is satisfied



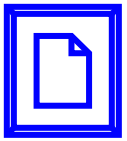
Synthesis

- For finite model can solve game using fixed point calculation
- Implementation in WIS model-checker TLV
- Main problem – complexity
- Practical Approach – User refines the LSCs and provides more details for solving the game

Synthesis Approach

- User provides more info by refining LSCs so (smart) play-out can win the game
- Play-out *(Harel & Marelly, SoSym '03)* - practical approach for executing LSCs – involves arbitrary choice between enabled events
- Smart Play-out *(FMCAD '02)* - model-checking methods make it possible to find a "correct" superstep if exists
- Synthesis – *(IJFCS '02)* Solving the full game

Synthesis Approach

- Tool provides support for refining LSCs so (smart) play-out is correct
- Sound approach – prove that we have a winning strategy – if succeed go on to generate statecharts
- Statechart Synthesis
  - Orthogonal states for different scenarios of same object
  - States correspond to progress in scenario
  - Communication using events (signals)
  - Primitive operations change attribute values

Demo