The Unheralded Value of the Multiway Rendezvous: Illustration with the Production Cell Benchmark

Hubert Garavel **Wendelin Serwe**

Inria Grenoble – LIG

Université Grenoble Alpes

http://convecs.inria.fr

Outline

- 1. The Multiway Rendezvous
- 2. The Production Cell Case Study
- 3. LNT Specification of a Production Cell Controller
- 4. Code Generation from the LNT Specification
- 5. Validation of the LNT Specification
- 6. Conclusion

1. The Multiway Rendezvous

What is multiway rendezvous?

- Binary rendezvous generalized to N>2 processes
- How does it work?
  - N processes execute asynchronously
  - they have to synchronize
  - processes that arrive early wait for other processes
  - once all processes are there, rendezvous occurs
  - data may be exchanged during rendezvous

    (in this talk: no data exchange, synchronization only)
  - after the rendezvous, each process resumes its execution asynchronously

History of the multiway rendezvous

1978	1980	1984	1988	1999	2001	2006-now
CSP	CCS	TCSP	LOTOS	LOTOS NT graphical parallel	E-LOTOS	LNT

- CSP: **rendezvous**
  - unifies communication and synchronization
  - binary, with named senders and receivers
- CCS: **ports** (or gates), SOS semantics
- TCSP: **multiway** rendezvous
- LOTOS: multiple values (*offers*), type checking, Boolean guards (conjunction of constraints)
- **Graphical *n*-ary** parallel composition operators

Discussion

- Powerful abstraction
  - one of the best features of LOTOS
- Similar concepts
  - naturally supported by Petri nets
  - mCRL2: also present (but output-only syntax)
  - synchronization barriers
  - synchronous languages
- Difficult to implement in a distributed setting
  - complex synchronization protocols are required
  - DLC compiler [Evrard et al.]

Uses of multiway rendezvous

- Observers
  - passively monitor without perturbing
  - examples: count messages, compute list of messages
- Supervisors
  - actively control/block actions
  - example: serialize actions
  - constraint-oriented specification style
- Atomic consensus
- Coordination of concurrent controllers (this paper)

2. The Production Cell Case Study

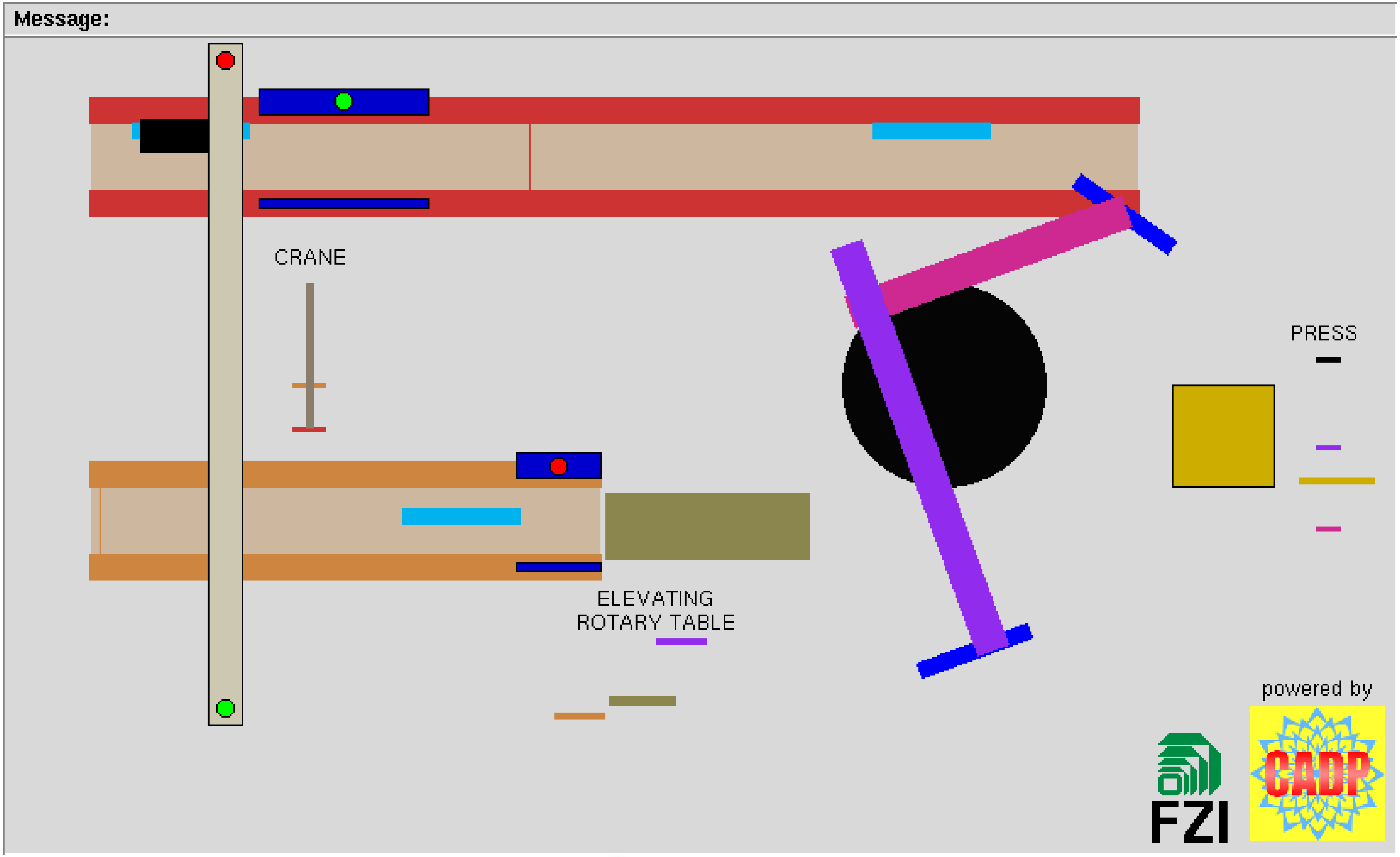
The Production Cell benchmark

- A famous case study of the mid 90s [LNCS 891]
- Goal: assessment of formal methods for the development of critical control software
- Replication of a metal-processing plant near Karlsruhe (Germany)
- Models in 30 different languages
  - see Appendix A of the paper
  - most specifications are not executable
  - only 5 papers report code generation and simulation
  - none of these 5 papers uses multiway rendezvous

Production Cell architecture

- 6 components:
  feed belt, elevating rotary table, rotating two-armed robot, press, deposit belt, and crane
- 14 sensors (S_1 to S_{14}):
  switches, potentiometers, and photoelectric cells
- 13 actuators (A_1 to A_{13}):
  motors and magnets
- Graphical simulator written in Tcl/Tk

Tcl/Tk simulator of the Production Cell



Tcl/Tk simulator of the Production Cell

- Control via character-string commands and replies
  - commands to control each actuator
  - single command to acquire values of **all** sensors
- Synchronous mode: infinite loop of reaction steps
  - acquire current sensor values (get_status)
  - compute appropriate reaction
  - send commands to the actuators
  - terminate reaction step (react)

3. LNT Specification of a Production Cell Controller

LOTOS and LNT models

- 1994: early LOTOS specifications
  - experiments with various architectures
  - problems to connect to the simulator
- 1997: LOTOS connected to the Tcl/Tk simulator
- 2013: translation to LNT
  - improvements of the LOTOS model
  - parallelisation of the commands
- 2017: further enhancements

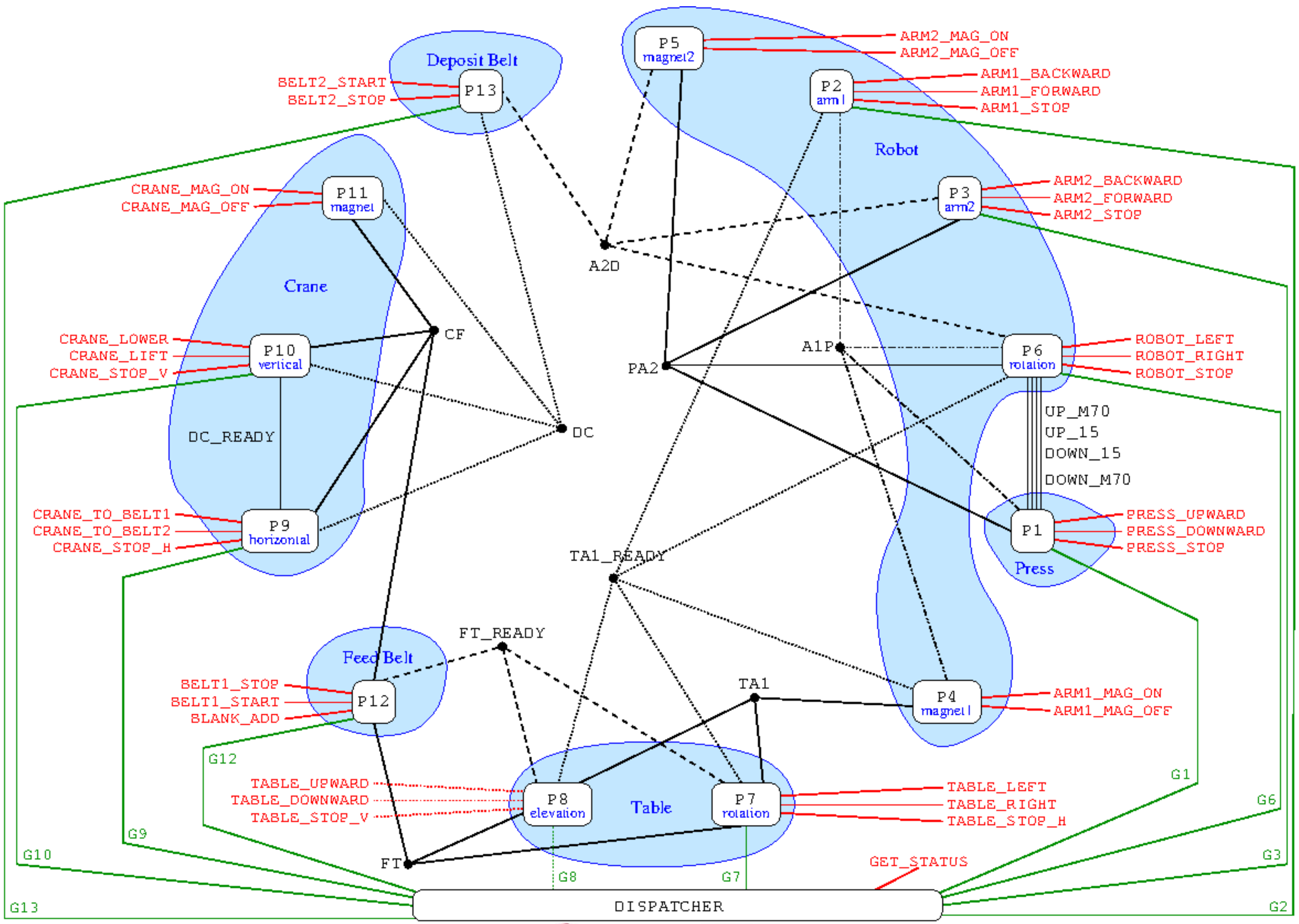
Controller architecture

- Decomposition following the production cell structure
- Control each degree of freedom separately
- Parallel composition with one process per actuator
- Each command modelled by a dedicated gate
  - single rendezvous for instantaneous transfers from the press to arm 2 of the robot: gate PA2
  - several rendezvous for transfers taking time from the feed belt to the table: gates FT_READY and FT (the belt must move during transfer)

Multiway rendezvous examples

- 2-way: DC_READY
  horizontal and vertical position of the crane ready for a transfer of an item from the deposit belt
- 3-way: FT
  transfer of an item from the feed belt to the table
- 4-way: PA2
  transfer of an item from the press to the 2nd robot arm (arm extension, magnet, rotation, and press)
- 5-way: TA1_READY
  table ready to transfer an item to 1st robot arm (two position of the table and three positions of arm 1)

Controller architecture



Dispatcher process

- Connect asynchronous controller with the synchronous Tcl/Tk simulator
- Simulator protocol:
  - access of all sensor values in a single step
  - dispatch relevant values to control processes
- Convert concrete sensor values to abstract ones
- Implementation choices
  - sequential dispatch in a predefined fixed order
  - parallel dispatch in an arbitrary order

Data aspects

- Sensor values: only basic types (Bool, Real, String)
- Floating-point precision: 10^{-2}
- Data abstractions:
  - keep only "extreme" values
    arm1’s extension: 0.5208, 0.6458, “other”
  - combine several sensors in one
    press position: “bottom”, “middle”, “top”, “other”

Individual processes Pi

- all individual processes have a similar structure:
  a cycle of actions, possibly with a initial sequence

```
process P11 [CRANE_MAG_ON, CRANE_MAG_OFF, DC, CF: NONE] is
   -- this process controls the magnet of the crane
   loop
      DC;
      CRANE_MAG_ON;
      CF;
      CRANE_MAG_OFF
   end loop
end process
```

Process P3

infinite cycle of 6 actions

synchr. on ARM2_STOP

read sensor S3 and enable ARM2_STOP if arm2 has its min. or max. value

```
process P3 [G3: ARM2_EXTENSION,
        ARM2_FORWARD, ARM2_STOP, ARM2_BACKWARD, PA2, A2D: NONE] is
  -- this process controls the extension of arm 2
  -- initially, arm 2 is completely retracted
  par ARM2_STOP in
    loop
      ARM2_FORWARD;
      ARM2_STOP; -- 0.7971
      PA2;
      ARM2_BACKWARD;
      ARM2_STOP; -- 0.5707
      A2D
    end loop
  ||
    var STATE: TWO_STATE,
        VALUE: ARM2_EXTENSION
    in
      STATE := 1;
      loop
        G3 (?VALUE);
        if LIMIT_ARM2_EXTENSION (STATE, VALUE) then
          ARM2_STOP;
          STATE := SUCC (STATE)
        end if
      end loop
    end var
  end par
end process
```

4. Code Generation from the LNT Specification

Code generation

- LNT ⟶ LOTOS ⟶ C code
- Sequential code generation
  - fully automated
- Need to connect this C code to the Tcl/Tk simulator
- How to connect a process calculus to a real system?
  - EXEC/CÆSAR framework for rapid prototyping
  - to each LNT gate, one associates a gate function
  - skeletons for gate functions automatic generated
  - some handwritten C code needed
  - main loop exploring an (infinite) execution path

5. Validation of the LNT Specification

Various checks

- Compile-time checks by LNT and LOTOS compilers
- Co-simulation with Tcl/Tk simulator (5 days)
- Many properties guaranteed by construction
  - avoid out-of-range movements
  - avoid collisions and dropping blanks
  - stop a motor before reversing its direction
  - at most one command per actuator/reaction step
- Complete verification remains to be done
  - state space explosion is challenging!

6. Conclusion

Conclusion

- Production cell is a stimulating benchmark
  - stable and precise specifications
  - involved Tcl/Tk simulator
- Multiway rendezvous enables a clean architecture
  - compositional and flexible
  - concurrent processes to control each degree of freedom separately
  - synchronize on goals of common interest
- Challenge: formal verification of the controller
  - model checking, equivalence checking, proofs ?