Distributed Games

Madhavan Mukund

Chennai Mathematical Institute
http://www.cmi.ac.in/~madhavan

Formal Methods Update 2017
IIT Mandi, 17 July 2017

Church's Problem

- Transform input bitstream into output bitstream

output

$\beta = 11010\ldots$

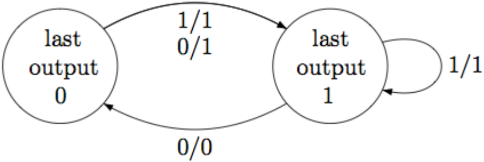
input

$\alpha = 01101\ldots$

- Input-output relationship is specified
- Synthesize a transducer that meets the specification
- $\beta(t)$ should be generated immediately after $\alpha(t)$v
  - In general, $\beta(t)$ could depend on $\alpha(0)\alpha(1)\cdots\alpha(t)$
  - Output 1 iff number of 0's in input so far is even

An Example

- Behaviour is specified in, say, MSO
  - $\forall t.\alpha(t) = 1 \Rightarrow \beta(t) = 1$
  - $\neg\exists t.\beta(t) = \beta(t+1) = 0$
  - $\exists^\omega t.\alpha(t) = 0 \Rightarrow \exists^\omega t.\beta(t) = 0$
- A 2-state strategy
  - On input 1, produce 1
  - On input 0, invert the last output



The result

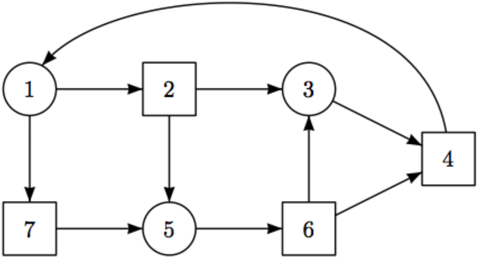
Büchi-Landweber Theorem (1969)

Any MSO specification can be implemented as a finite-state transducer.

From synthesis to games

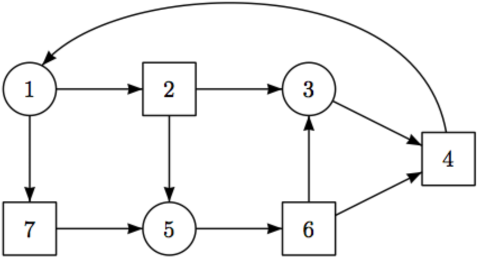
- Church's Problem: Synthesis or Realizability
- Alternatively, view as a 2-player game
  - Player A generates $\alpha(t)$
  - Player B generates $\beta(t)$
  - Player B wins if $(\alpha(0)\alpha(1)\cdots, \beta(0)\beta(1)\cdots)$ meets the specification
- Constructing a winning strategy for B solves the synthesis problem

Games on graphs



- Move a token around the graph
- Square nodes belong to A, circles to B
- Player who owns the node makes the next move

Winning conditions



- Reach node 3
  - *A* wins unless game starts in node 3
  - Divert the token from 2 to 5 and from 6 to 4
- Visit $\{2, 7\}$ infinitely often
  - From 1, *B* alternately moves to 2 and 7

From synthesis to games

Büchi-Landweber Theorem (1969)

Any MSO specification can be implemented as a finite-state transducer.

- Can transform MSO specifications into automata
- Synthesis problem becomes a graph game on the underlying automaton
- Finite-state strategy for graph games solves the synthesis problem
- Alternative proof of Büchi-Landweber Theorem

From realizability to controllability

- Synthesis or realizability asks to construct an automaton that implements a specfication
- Controllability asks to restrict the behaviour of a given automaton
- Closed system — automaton in which all transitions can be chosen by the system
- Open system — some transitions are decided by the environment, uncontrollable

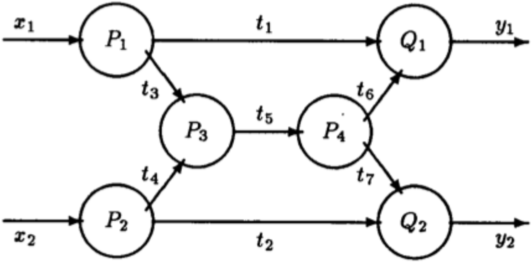
Controlling Discrete Event Systems

- Ramadge and Wonham, 1989
- Partition actions Σ as controllable Σ_c and uncontrollable Σ_u
- Given transition system generates a prefix closed language L
- Want to constrain the behaviour within $K \subseteq L$
- If $wa \in L$ and $a \in \Sigma_c$, supervisor can disable a
- If $w\alpha \in L$ and $\alpha \in \Sigma_u$, supervisor cannot disable α
- For regular specifications, can synthesize a finite-state supervisor

The distributed setting

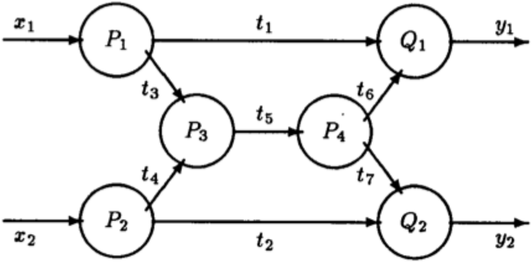
- Pnueli and Rosner (1989)
- Generalize to multiple interconnected processes
- Distributed inputs, outputs and intermediate shared variables
- Again, address realizability and controllability

Distributed architecture



- Inputs x_i, outputs y_j, shared communication variables t_k
- Underlying graph is acyclic
- Each process P has input and output variables $in(P)$ and $out(P)$
- P implements a local strategy to compute $out(P)$ at t from $in(P)$ at $0, 1, \ldots, t$

Distributed synthesis



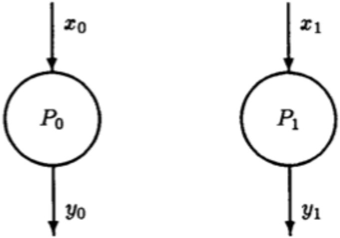
- Given a specification over inputs and outputs $\varphi(\bar{x}, \bar{y})$ implement it over a compatible architecture (distributed implementability)
  - Trivial solution uses a single process
- Given a specification $\varphi(\bar{x}, \bar{y})$ and an architecture $\mathcal{A}$, find an implementation over $\mathcal{A}$ (distributed realizability)

Distributed realizability

Pnueli-Rosner (1989)

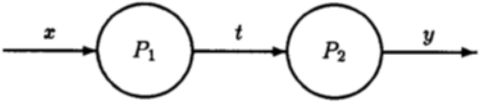
Distributed realizability is undecidable.

- Disconnected architecture
- Components simulate the tape of a Turing machine
- Global specification combining both inputs and outputs can be realized iff the given Turing machine halts

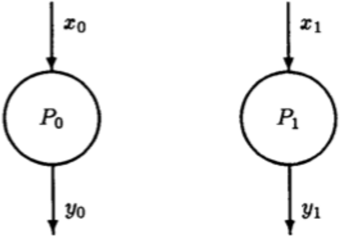


Distributed realizability

- The single node architecture is decidable (Büchi-Landweber)
- The only decidable architectures are pipelines . . .



- In the undecidability proof, the specification links $\{x_0, y_0\}$ and $\{x_1, y_1\}$ though there is no communication
- "Local" specifications?



Controllability with local specifications

- Madhusudan and Thiagarajan, 2001
- Specification for each process in terms of its inputs and outputs
- Study controllability rather than realizability
- Slight generalization of decidable architectures to "clean" pipelines

Distributed alphabets

- Actions $\Sigma = \{a, b, \ldots\}$
- Processes $\mathcal{P} = \{p, q, \ldots\}$
- Each action a has a set of readers, $R(a)$, and a set of writers, $W(a)$
  - $W(a) \subseteq R(a)$
  - $R(a) \cap W(b) = \emptyset$ *iff* $R(b) \cap W(a) = \emptyset$
- An a-transition reads states of processes in $R(a)$ and updates states of processes in $W(a)$
- Special case is usual distributed alphabet — $loc(a) = R(a) = W(a)$ for each a

Controllability

- Partition Σ as controllable Σ_c and uncontrollable Σ_u
- Each action a has an underlying transition relation Δ_a
- For an action $a \in \Sigma_c$, agents can choose a transition from Δ_a
- For an action $b \in \Sigma_u$, environment can choose any transition from Δ_b
- Winning condition
  - In terms of global states observed across all subtraces
- Control problem
  - Come up with a strategy to guide controllable transitions to achieve the winning condition

Local strategies

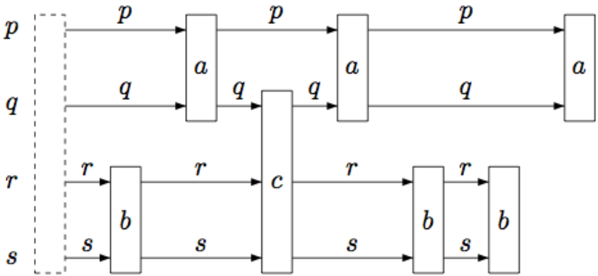
- A strategy decides the next action based on the past history
- What history should be observable?
- The Pnueli-Rosner undecidability result shows that access to global information is unreasonable
- How does one define local information?

Distribution and independence

- Two actions are independent if they cannot influence each other
- If $R(a)$ overlaps with $W(b)$, occurrence of b can change options for a
- Define a and b to be independent if $R(a)$ is disjoint from $W(b)$
- The constraint $R(a) \cap W(b) = \emptyset \textit{ iff } R(b) \cap W(a) = \emptyset$ ensures that this is a symmetric relation
- Independence relation $I \subset \Sigma \times \Sigma$— symmetric, irreflexive
- Dependence relation, complement, $(\Sigma \times \Sigma) \setminus I$— symmetric, reflexive

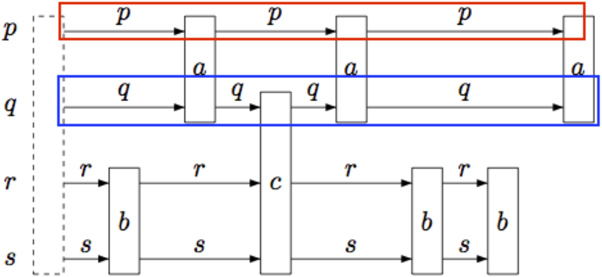
Mazurkiewicz Traces

- Independence imposes a labelled partial order structure on runs
- Mazurkiewicz trace



Strictly local history

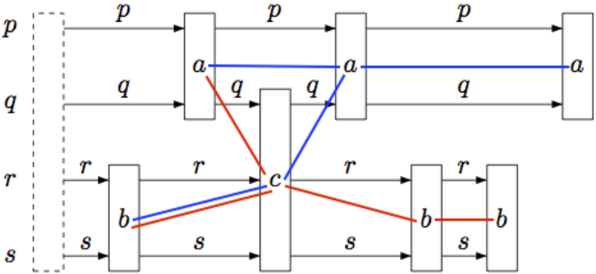
- A process can see the actions in which it took part



- Does not account for information communicated through synchronizations

Causal history

- A process can see the actions which it has heard about
  - Directly, it is part of $W(a)$
  - Indirectly, it has information through partners of other actions



Causal memory strategies

- Control strategy has access to causal history
- Which games are decidable using such strategies?
- Can these strategies be implementing as finite-state?
  - Remember only a bounded amount of the causal past

Characterizing architectures

- Consider the dependency graph (Σ, D)
  - Vertices are actions
  - Edges are dependence relation
- Structure of dependency graph is a way to measure the complexity of the distributed architecture
- Look at special cases
  - Series-parallel — dependency graph built by a sequence of sequential and parallel composition operations
  - Trees

What is known

- Series-parallel graphs [Gastin, Lerman and Zeitoun, 2004]
  - Causal memory strategy implies a bounded memory strategy
  - Existence of causal memory strategy is decidable
- Tree architectures [Genest, Gimbert, Muscholl, Walukiewicz, 2012]

What is not known

- Is the existence of causal memory strategies decidable for all architectures?