Semi Autonomous Driving

Prafulla Saxena , Vasu Bansal , I G Prasad

IIT Kanpur

3rd April

Outline

Introduction

Human in the loop Control Systems :

- A control system which involves the interaction of an autonomous controller with one or more human operators
- Examples : Autopilot System with pilot interaction, Automated driver assistance, Autonomous driving with driver interaction etc.

Introduction

Why Formal Methods for HuIL Systems:

- Cost associated with incorrect operation can be very severe
- May even cost a life
- Can we verify if autonomous controller is correct?
- Correctness depends also on the actions taken by human controller.
- Solution ? : Fully autonomous System. However existence is not always guaranteed.
- But fully mannual control puts too much burden on the operator. HuIL combines both autonomous and human effort.
- Systematic method to synthesize such a combination is required.

Problem Statement

Can we devise a controller that is mostly automatic and requires occasional human interaction for correct operation?

- Here we are considering one of the interesting domain, which is 'self-driving' or 'semi-autonomous cars' with mode(Level 3) of automation as defined by NHTSA.

"Level 3 - Limited Self-Driving Automation: Vehicles at this level of automation enable the driver to cede full control of all safety-critical functions under certain traffic or environmental conditions and in those conditions to rely heavily on the vehicle to monitor for changes in those conditions requiring transition back to driver control. The driver is expected to be available for occasional control, but with sufficiently comfortable transition time. The vehicle is designed to ensure safe operation during the automated driving mode." [13]

Criteria Required for HuIL

- **Monitoring** - monitor the past/current info to determine human intervention.
- **Minimal Intervention** - Invoke human only if necessary (minimally)
- **Prescient** - Determine if something may go wrong ahead of time.
- **Conditionally Correct** - correct operation until human intervention is found to be necessary.

Components Overview

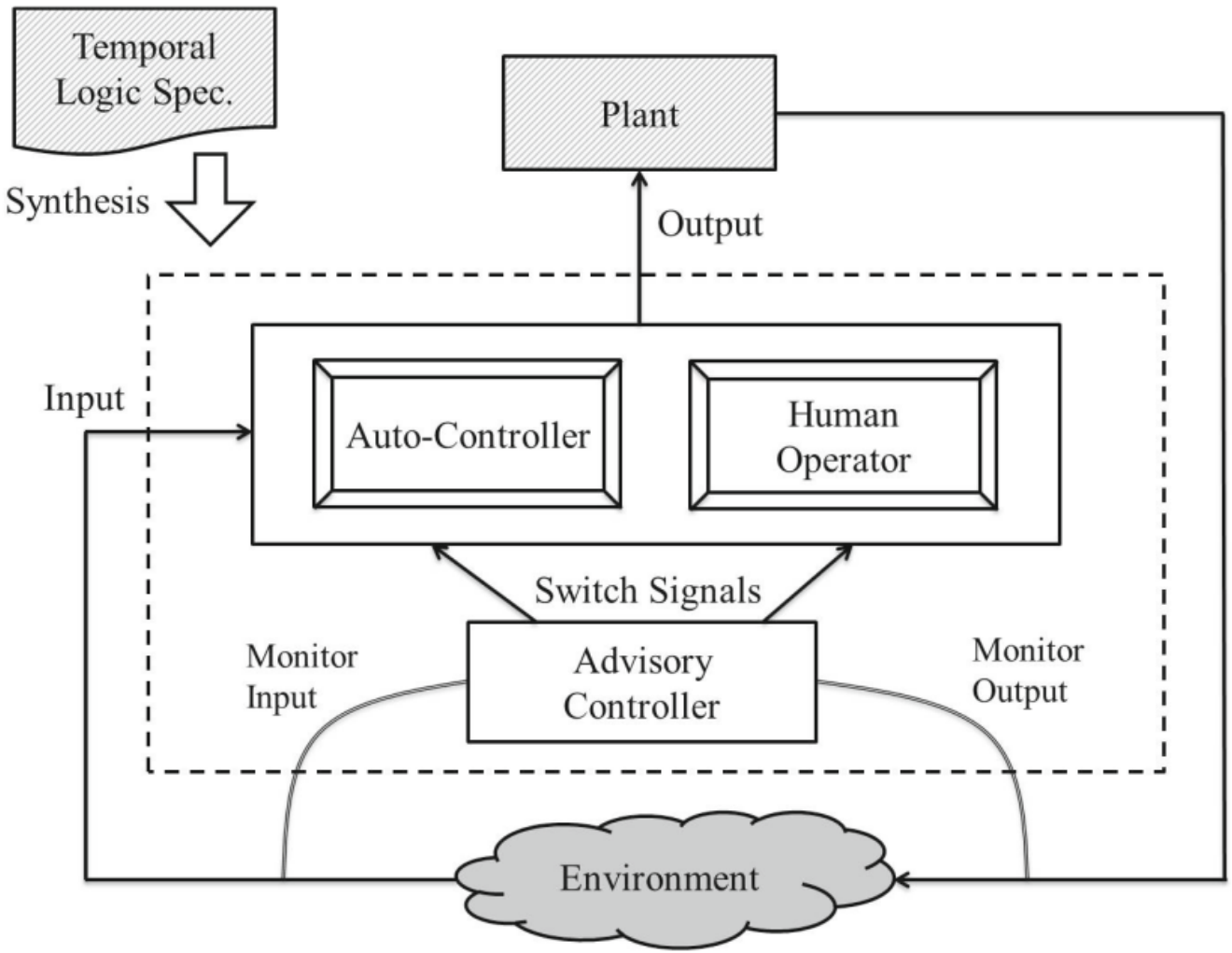
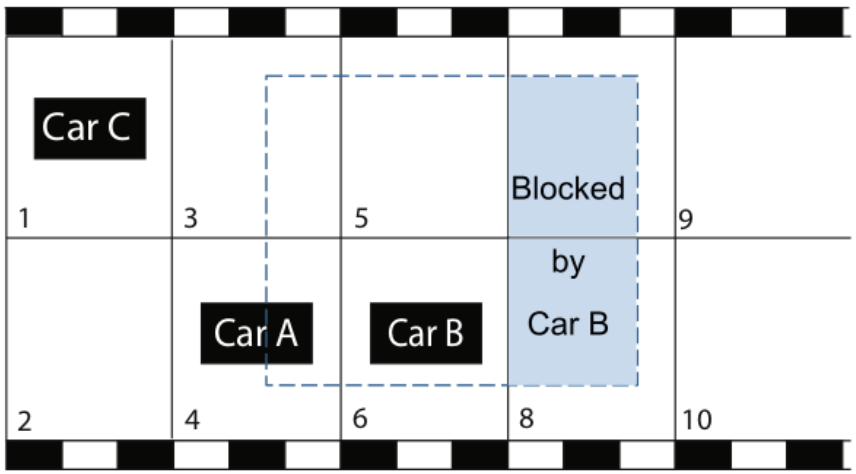
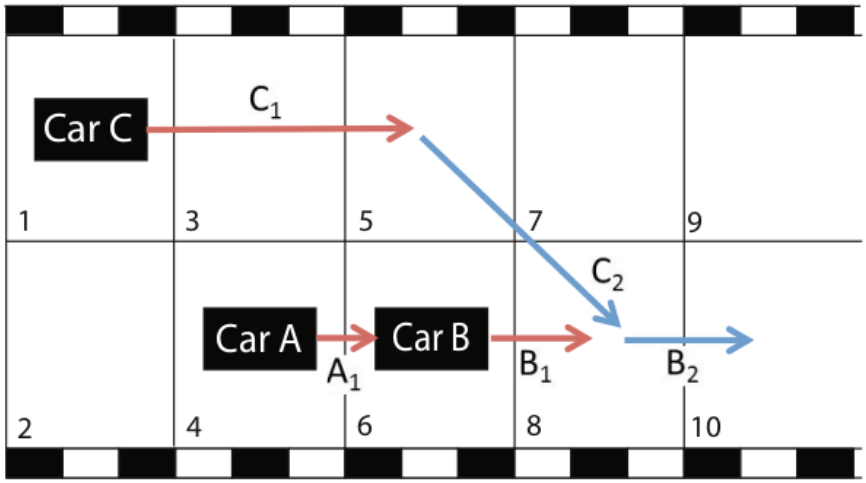


Fig. 1. Human-in-the-Loop Controller: Component Overview

Motivating Example



(a) A's Sensing Range.



(b) Failed to Follow.

Fig. 2. Controller Synthesis – Car A Following Car B

Motivating Example

Goal - Automatically synthesize a controller for car A such that:

- car A follows car B whenever possible;
- when the objective may not be achievable, switches control to the human driver while allowing sufficient time for the driver to respond and take control

Assumption -

- Car A sensor can see two block forward
- A and C can move at most 2 squares forward, but car B can move at most 1 square ahead, otherwise car B can out-run car A

Formal Model of HuIL Controller

In this paper they model a discrete controller as a finite-state transducer(Mealy machine)

Notation

- $M = (Q, q_0, \mathcal{X}, \mathcal{Y}, \rho, \delta)$
- Q = set of states, q_0 initial state
- $\mathcal{X} = 2^X, \mathcal{Y} = 2^Y$ is Booleanized space, where X and Y are two disjoint sets of variables representing inputs and outputs respectively.
- $\rho : Q \times \mathcal{X} \rightarrow Q$ Transition function
- $\delta : Q \times \mathcal{X} \rightarrow \mathcal{Y}$ Output function
- Function $\mathcal{F} : Q \rightarrow \{true, false\}$ characterize correctness of M. *if* $\mathcal{F}(q) = true$ then state q is "Failure prone"
- A run of M $q = q_1 q_2 .....$ such that $q_{k+1} = \rho(q_k, i_k)$
- Word produced $M(x) = \delta(q_0, x_0)\delta(q_1, x_1)$

Agents in Human-in-loop controller

- The overall controller HuIL is a composition of the 3 models of Auto controller($\mathcal{AC}$), Human controller($\mathcal{HC}$) and Advisory controller($\mathcal{VC}$)
- A binary variable **auto** is internal advisory signal, controlled by $\mathcal{VC}$ *if* `auto = false`, it means $\mathcal{HC}$ is required to take over control
- A binary variable **active** is used to denote if $\mathcal{HC}$ is active or not. The "overwrite" action happens over output when human operator is in control.
- To minimize the need to engage human operator, a **Joint objective function** $\mathcal{C}$ is used.
- This optimise the trade off between human cost and failure probability of $\mathcal{AC}$ which allows human oprator to take control before time T of any failure.
- This allows to successfully derive four criteria given by NHTSA.

Synthesis from Temporal Logic

- **Realizability :** determining whether there exists a transducer M with input $\mathcal{X} = 2^X$ and output $\mathcal{Y} = 2^Y$ such that $M \models \psi$. Where ψ is LTL formula which need to be satisfy.
- Complexity of deciding the realizability of an LTL formula can be prohibitively high (2EXPTIME-complete)
- Can use Generalized Reactivity(1) [GR(1)] more efficient algorithm of synthesizing.
- Has form $\psi = \psi^{env} \rightarrow \psi^{sys}$

- ψ_i^l: a Boolean formula that characterizes the *initial states*.
- ψ_t^l: an LTL formula that characterizes the *transition*, in the form $\mathbf{G}\ B$, where B is a Boolean combination of variables in $X \cup Y$ and expression $\mathbf{X}\ u$ where $u \in X$ if $l = env$ and $u \in X \cup Y$ if $l = sys$.
- ψ_f^l: an LTL formula that characterizes *fairness*, in the form $\mathbf{G}\ \mathbf{F}\ B$, where B is a Boolean formula over variables in $X \cup Y$.

Games and Strategies

A finite-state two-player game is defined by :

- $\mathcal{G} = (Q^g, \theta^g, \rho^{env}, \rho^{sys}, W_{in})$
- $Q^g \subseteq 2^{X \cup Y}$
- θ^g is a Boolean formula over $X \cup Y$ specifies the initial states.
- $\rho^{env} \subseteq Q^g \times 2^X$ environment transition relation
- $\rho^{sys} \subseteq Q^g \times 2^X \times 2^Y$ system transition relation
- W_{in} is the winning condition
- $\theta^g = \psi_i^{sys} \wedge \psi_i^{env}$
- $\rho^{env} = \psi_t^{env}$ and $\rho^{sys} = \psi_t^{sys}$
- $W_{in} = \psi_f^{env} \rightarrow \psi_f^{sys}$
- A play π is winning for the system iff it is infinite and $\pi \models W_{in}$

Games and Strategies

A finite-memory strategy for env in $\mathcal{G}$ is a tuple:

- $S^{env} = (\Gamma^{env}, \gamma^{env}, \eta^{env})$
- Γ^{env} finite set representing the memory
- $\gamma^{env_0} \epsilon \gamma_{env}$ initial memory content
- $\eta_{env} \subseteq Q^q \times \Gamma^{env} \times \mathcal{X} \times \Gamma^{env}$ is a relation mapping a state in $\mathcal{G}$ and some memory content $\gamma^{env} \epsilon \Gamma^{env}$ to the possible next inputs the environment can pick and an updated memory content
- The existence of a counter-strategy is equivalent to the specification being unrealizable

Counter-strategy Graph

A counter-strategy graph G^c is a discrete transition system

- $G^c = (Q^c, Q_0^c \subseteq Q^c, \rho^c \subseteq Q^c \times Q^c)$
- $Q^c \subseteq Q^g \times \Gamma^{env}$ is the state space
- $Q_0^c = Q_0^g \times \gamma^{env_0}$ is the set of initial states
- $\rho^c = \eta^{env} \wedge \rho^{sys}$ is the transition relation
- For convenience, they use a function $\theta^c : Q^c \rightarrow 2^{X \cup Y}$ to denote the game state.

Failure scenario

- **Safety violation** : For a node say q_1^c if there does not exist a node q_2^c such that $(q_1^c, q_2^c)\,\epsilon\,\rho^c$ then q_1^c is 'Failure imminent'
- System unable to find the next output such that all safety guarantees are satisfied.
- **Fairness violation** : If a node q^c is part of a strongly connected component (SCC) in Q^c, then q^c is failure-doomed.
- env can always pick an input so play is forced to get stuck in scc

G^c to DAG

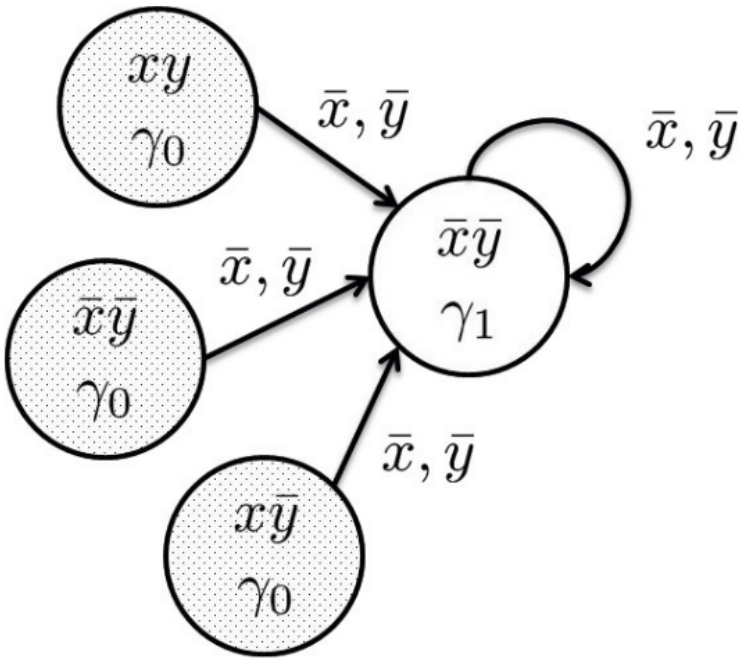
Can convert Counter-strategy Grap (G^c) to Directed Acyclic Graph (DAG) by contracting each SCC in G^c to a single node

- $\hat{G}^c = (\hat{Q}^c, \hat{Q}_0^c, \hat{\rho}^c)$
- $\hat{f} : Q^c \rightarrow \hat{Q}^c$ is Surjective function to describe mapping of nodes from G_c to $\hat{G}_c$

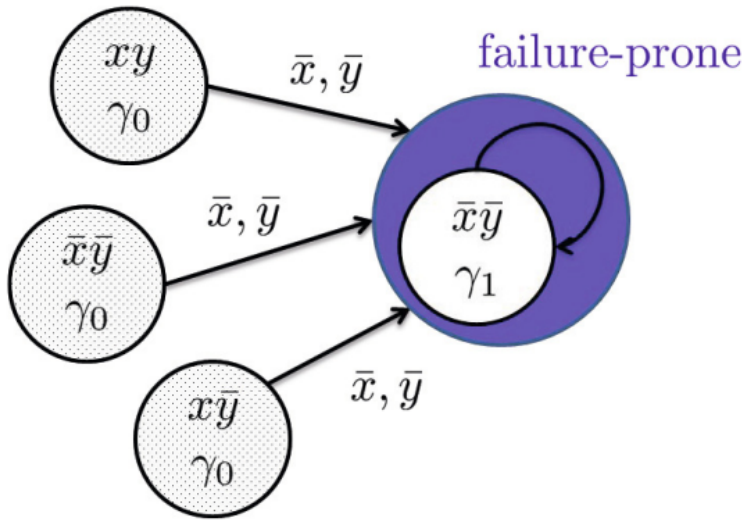
Example

Example 1. Consider $X = \{x\}, Y = \{y\}$ and the following GR(1) sub-formulas which together form $\psi = \psi^{env} \rightarrow \psi^{sys}$.

1. $\psi_f^{env} = \mathbf{G}\ (\mathbf{F}\ \neg x)$
2. $\psi_t^{sys} = \mathbf{G}\ (\neg x \rightarrow \neg y)$
3. $\psi_f^{sys} = \mathbf{G}\ (\mathbf{F}\ y)$



(a) Counterstrategy graph G^c for unrealizable specification ψ.



(b) Condensed graph $\hat{G}^c$ for G^c after contracting the SCC.

Weight Assignment and cost computation $\hat{G}^c$

$$\varpi(\hat{q}_i, \hat{q}_j) = \begin{cases} 1 & \text{if } \hat{q}_j \text{ is } \textit{failure-prone} \\ \frac{pen(\hat{q}_i) \times len(\hat{q}_i)}{c(\hat{q}_i)} & \text{Otherwise} \end{cases}$$

Cost function: $f_C = \Sigma_{e \epsilon E^\phi} \varpi(e)$

- $\varpi(\hat{q}_i, \hat{q}_j) =$ weight assigned to edge
- $c(q_i)$ is total number of legal actions env can take from q_i
- $len(q_i) : Q^c \rightarrow Z^+$ is the number of edges in shortest path from a node q_i to any failure-prone node
- $pen(q_i)$: user-defined penalty parameter should be chosen such that $\varpi(\hat{q}_i, \hat{q}_j) < 1$
- E^ϕ : set of edges whose removal avoids the reach-ability of any failure-prone state from initial state.

Counter-Strategy Guided Synthesis

- Goal : Eliminate counter-strategy
- Need to assert the negation of the corresponding conditions (which responsible for violation) as additional environment transition assumptions.
- Need to mine assumptions $\phi = \wedge_i(\mathbf{G}(a_i \rightarrow \neg \mathbf{X} b_i))$
  a_i : is a Boolean formula describing a set of assignments over variables in $X \cup Y$
  b_i : is a Boolean formula describing a set of assignments over variables in X.
- $(\mathbf{G}(x \wedge y \rightarrow \neg \mathbf{X}\bar{x})) \wedge (\mathbf{G}(\bar{x} \wedge \bar{y} \rightarrow \neg \mathbf{X}\bar{x})) \wedge (\mathbf{G}(x \wedge \bar{y} \rightarrow \neg \mathbf{X}\bar{x}))$
- Under the assumption ϕ, if $(\phi \wedge \psi^{env}) \rightarrow \psi^{sys}$ is realizable, then we can automatically synthesize an auto-controller that satisfies ψ
- Basically this is finding a set of edges called E^{ϕ}

Counter-Strategy Guided Synthesis

- Criteria *Prescient* comprises that no failure-prone nodes is reachable from a non-failure-prone node within less than T steps.
- Notion of *Minimally intervening* taken care by minimising the Cost function $\Sigma_{e \epsilon E^{\phi}} \varpi(e)$
- Given $\hat{G}^c$, we compute $\hat{Q}^c_T \subseteq \hat{Q}^c$ which are backward reachable within $T - 1$ steps from any failure-prone node
- Remove these nodes and obtain a new DAG $\hat{G}^c_T$
- Now we can formulate a s-t min cut problem (by adding new source and sink node) to find a minimum cut.

Algorithm

Algorithm 1. Counterstrategy-Guided HuIL Controller Synthesis

Input: GR(1) specification $\psi = \psi^{env} \rightarrow \psi^{sys}$.
Input: T : parameter for minimum human response time.
Output: $\mathcal{AC}$ and $\mathcal{VC}$. $\mathcal{HUIL}$ is then a composition of $\mathcal{AC}$, $\mathcal{VC}$ and $\mathcal{HC}$.

if ψ is *realizable* **then**
  Synthesize transducer $M \models \psi$ (using standard GR(1) synthesis);
  $\mathcal{HUIL} := M$ (fully autonomous).
else
  Generate G^c from ψ (assume a single G^c; otherwise the algorithm is performed iteratively);

  Generate the DAG embedded $\hat{G}^c$ from G^c.
  Reduce $\hat{G}^c$ to $\hat{G}^c_T$;
  Assign weights to $\hat{G}^c$ using φ; by removing $\hat{Q}^c_T$ – nodes that are within $T-1$ steps of any failure-prone node;
  Formulate a *s-t* min-cut problem with $\hat{G}^c_T$;
  Solve the *s-t* min-cut problem to obtain E^ϕ;
  Add assumptions ϕ to ψ to obtain the new specification $\psi_{new} := (\phi \wedge \psi^{env}) \rightarrow \psi^{sys}$;
  Synthesize $\mathcal{AC}$ so that $M \models \psi_{new}$;
  Synthesize $\mathcal{VC}$ as a (stateless) monitor that outputs $auto =$ `false` if ϕ is violated.
end if

Experimental Results

Consider car-following example,

- Any position can be occupied by at most one car at a time (no crashing):

$$\mathbf{G}\left(p_A = x \rightarrow (p_B \neq x \wedge p_C \neq x)\right)$$

- Car A is required to follow car B:

$$\mathbf{G}\left((v_{AB} = \texttt{true} \wedge p_A = x) \rightarrow \mathbf{X}\left(v_{AB} = \texttt{true}\right)\right)$$

- Two cars cannot cross each other if they are right next to each other:

$$\mathbf{G}\left(((p_C = 5) \wedge (p_A = 6) \wedge (\mathbf{X}\, p_C = 8)) \rightarrow (\mathbf{X}\left(p_A \neq 7\right))\right)$$

Experimental Results

By applying algorithm to this (unrealizable) specification with T = 1, this will give following assumption ϕ

$$\phi = \mathbf{G}\left(((p_A = 4) \wedge (p_B = 6) \wedge (p_C = 1)) \rightarrow \neg\mathbf{X}\left((p_B = 8) \wedge (p_C = 5)\right)\right) \bigwedge$$

$$\mathbf{G}\left(((p_A = 4) \wedge (p_B = 6) \wedge (p_C = 1)) \rightarrow \neg\mathbf{X}\left((p_B = 6) \wedge (p_C = 3)\right)\right) \bigwedge$$

$$\mathbf{G}\left(((p_A = 4) \wedge (p_B = 6) \wedge (p_C = 1)) \rightarrow \neg\mathbf{X}\left((p_B = 6) \wedge (p_C = 5)\right)\right)$$

Conclusion

- Propose a synthesis approach for designing human-in-the-loop controllers
- Propose an algorithm based on mining monitorable conditions from the counterstrategy of the unrealizable specifications.

Future scope

- They have not associated any probabilities with transition taken by env or sys. can be adapted in future work.
- Can Consider more complex human driver model as there is no direct control on human operator.
- when and how an auto controller take back the control.

References

- Wenchao Li, Dorsa Sadigh, S. Shankar Sastry, Sanjit A. Seshia: Synthesis for Human-in-the-Loop Control Systems. TACAS 2014: 470-484
- Piterman, N., Pnueli, A., Sa'ar, Y.: Synthesis of reactive(1) designs. In: Emerson, E.A.,Namjoshi, K.S. (eds.) VMCAI 2006. LNCS, vol. 3855, pp. 364–380. Springer, Heidelberg (2006)