Simple Synthesis of Reactive Systems with Tolerance for Unexpected Environmental Behavior

Shigeki Hagihara[1] Atsushi Ueno[2] Takashi Tomita[3]
Masaya Shimakawa[1] Naoki Yonezaki[4]

[1]Tokyo Institute of Technology

[2]Nintendo Co., Ltd.

[3]Japan Advanced Institute of Science and Technology

[4]The Open University of Japan

FormaliSE'16

Timing critical systems in our society

- Ex. systems controlling
  - elevators,
  - vending machines,
  - nuclear power plants, and
  - air traffic.
- Most of these systems are open systems which respond to requests from the environment at an appropriate time.
- Unlike general business software, fatal accidents cause severe damages, and it is important to ensure safety of the system.
  - Such systems became complex. it became unmanageable to develop the systems without flaws.
- One of the solutions is to use tools based on formal methods.
  - Recently, formal methods have become essential tools for developing safety critical open systems.

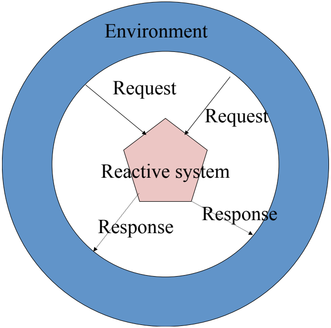
Formal methods for developing safety critical open systems

- Model checking
  - a method for checking whether models of systems satisfy specifications,
  - popular technique for developing practical systems such as systems controlling car, airplane, and rockets.
- System synthesis from specification
  - a method for checking whether there exists a system that satisfies specifications and synthesizing transition systems representing the system if there exists
  - it is not necessary to design systems manually, and systems which satisfy specifications can be synthesized automatically.

In this work, we focus on synthesis of open systems.

Reactive System

A formalization of open systems.



RS is a triple $\langle X, Y, r\rangle$, where

- X is a set of events caused by an environment,
- Y is a set of events caused by the system and
- $r : (2^X)^* \mapsto 2^Y$ is a reaction function.

Realizability

Reactive system specifications must satisfy realizability.

Realizability there exists a reactive system such that for any environmental events of any timing, the reactive system produces system events such that the specification holds.

Definition (Realizability)

A specification φ is *realizable* if the following holds:

$$\exists RS \forall \tilde{i} (behave_{RS}(\tilde{i}) \models \varphi),$$

where $\tilde{i}$ is an infinite sequence of sets of environmental events, i.e., $\tilde{i} \in (2^I)^\omega$. $behave_{RS}(\tilde{i})$ is the infinite behavior by RS for $\tilde{i}$.

Complexity

- Realizability problem is 2EXPTIME-complete [PR89, Ros92].

Tools for Realizability-checking and System synthesis

- Several tools have been proposed for checking realizability of specifications written LTL
  - Lily
  - AcaciaPlus
  - Unbeast
  - Our implementation (No name currently)

These tools can synthesize reactive system which is an evidence of realizability if the specification is realizable.

Example: Synthesis of a control system for a door

Example (A specification of a control system for a door)

- If the open button is pushed, the door opens at the next time unit.
  $$\varphi_{sys1} = G(open \rightarrow Xd)$$
- If the close button is pushed, the door eventually closes.
  $$\varphi_{sys2} = G(close \rightarrow XF\neg d)$$

(environmental events) $open, close$: ‘the open or close button is pushed,’

(system event) d:the door is open.

Example: A control system for a door (cont.)

The specification of this example is not realizable.

- Let us consider the case that the close button is held (i.e., pushed continuously) after the open button is pushed.
  - if the system eventually opens the door, the behavior contradicts φ_{sys2},
  - if the system always closes the door, the behavior contradicts φ_{sys1},

  $\Rightarrow$ The collision of φ_{sys1} and φ_{sys2} results in φ being unrealizable.

By adding the following assumption to the specification, the specification become realizable, i.e. $\varphi_{env} \rightarrow \varphi_{sys1} \wedge \varphi_{sys2}$ is realizable.

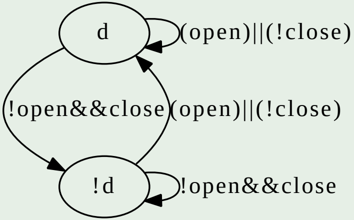
Assumption about environmental behavior

The buttons cannot be pushed simultaneously due to a physical constraint.

$$\varphi_{env} = G(\neg open \vee \neg close)$$

Synthesis from the specification of the door

A synthesized reactive system



This transition system is assured to satisfy the specification of the door.

Safraless synthesis method [KV05] (refined version [SF07, FJR09])

(1) We construct a universal coBüchi automaton A_φ, which only accepts the set of behaviors that satisfies φ.

(2) $k = 0$ and we iterate the following steps (a) and (b).

(a) We regard A_φ as a k-coBüchi automaton, and determinize it to obtain a safety game SG_k.

(b) Then we compute a winning region WR_k on SG_k. If the initial state is included in WR_k, we go to step (3), otherwise $k = k + 1$ and we return to (a).

(3) We derive a strategy from WR_k as a resulting reactive system.

Motivation

- From $\varphi_{env} \rightarrow \varphi_{sys}$,
  - we can synthesize a reactive system, whose behavior satisfies φ_{sys} as long as the behavior of the environment satisfies φ_{env}.
  - However, if the behavior of the environment does not satisfy φ_{env}, there is no restriction on the behavior of the synthesized reactive system, meaning that it may do nothing.
- In a real-world setting, it is often difficult to ensure that the behavior of the environment satisfies the desired constraints for the environment,

It is desirable for reactive systems to behave in such a way as to satisfy the given constraint φ_{sys} as much as possible, even if the behavior of the environment does not satisfy φ_{env}.

Goal

We provide a simple definition of environmental tolerance, and propose a method for synthesizing reactive systems with environmental tolerance based on our definition.

- In our synthesis method, we devise a way to derive a winning strategy from a winning region.

Towards defining Environmental tolerance

Let RS be a realization of $\varphi_{env} \rightarrow \varphi_{sys}$.

Intention of environmental tolerance

- Even for environmental behavior not satisfying φ_{env}, RS behaves in such a way as to satisfy φ_{sys} as much as possible.

- Several definitions for environmental tolerance were proposed in [BCG+14] and [Ehl11]. In these works, environmental tolerance was defined by using the ratio of the number of violations of φ_{sys} to the number of violations of φ_{env}.

We give a simpler quantitative definition for environmental tolerance, where only the number of violations of φ_{sys} is considered. To define this, we use the concept of mean-payoff to represent how often the behavior of RS satisfies φ_{sys}.

Payoff terms, Mean-payoff terms [THHY12]

A mean-payoff expression in LTL$^{\mathrm{mp}}$ represents *quantitative temporal properties*, *e.g.*.

Term	Meaning
$\mathrm{MP}(t)$	The *limit inferior of mean-payoffs* under a *payoff setting* t.
$\mathbf{1}_\psi$	Payoff 1 is gained for every step satisfying ψ.

- $[\![\mathrm{MP}(\mathbf{1}_{e_1})]\!]_\sigma \geq 1/2$:
  - the long-run frequency of event e_1 is bounded below by $1/2$.
- $[\![\mathrm{MP}(-2 \cdot \mathbf{1}_{e_2} - 10 \cdot \mathbf{1}_{e_3})]\!]_\sigma \geq -4$:
  - the long-run average of costs is bounded above by 4, when it costs 2 and 10 for each occurrences of events e_2 and e_3, respectively.

Defining environmental tolerance

For $\varphi_{sys} = G\psi_1 \wedge \ldots \wedge G\psi_n$, we obtain a payoff term $t_{\varphi_{sys}} = \sum_i c_i \cdot 1_{\psi_i}$,

Definition (Degree of environmental tolerance)

Let RS be a reactive system, φ_{sys} be a system constraint for RS, and $t_{\varphi_{sys}}$ be a payoff term. The degree of environmental tolerance of RS for $t_{\varphi_{sys}}$ is defined as follows.

$$Tol(RS, t_{\varphi_{sys}}) = \min_{\tilde{i} \in (2^I)^\omega} [\![MP(t_{\varphi_{sys}})]\!]_{behave_{RS}(\tilde{i})}$$

If $Tol(RS, t_{\varphi_{sys}})$ is a large value, we call RS a reactive system with environmental tolerance.

Proposed synthesis method

Let $\varphi = \varphi_{env} \to \varphi_{sys}$ be a specification, and t be a payoff term obtained from φ_{sys}.

(1) Based on the refined Safraless synthesis method, we construct WR_k from φ, where k is a number that obtains realization.

(2) We construct mean-payoff game MG_t that can be considered equivalent to t.

(3) We compute the synchronized product of WR_k and MG_t and obtain mean-payoff game $MG_{(t,\varphi)}$.

(4) We compute the optimal winning strategy of $MG_{(t,\varphi)}$ and output it.

Approximating until formulae

We restrict any temporal operators occurring in any formula in t to be the X operator.

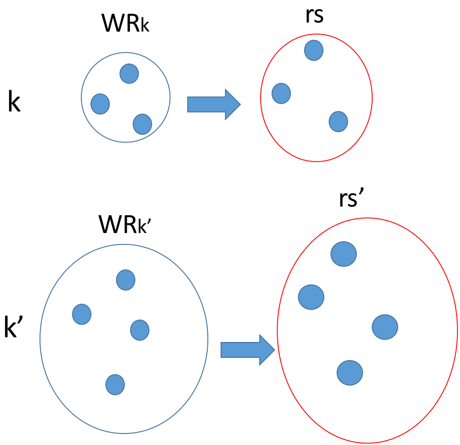
- 'Until' formula $\Longrightarrow$ bounded 'Until' formula
  If we use the boundary $n = 2$ for the bounded Until operator, we use the following formula with the X operator instead of fUg.

$$fUg \approx fU^2g = g \vee (f \wedge Xg) \vee (f \wedge Xf \wedge XXg)$$

Constructing mean-payoff game

- Since t_ψ contains an operator X only, such as $\psi = a \to (Xb \vee XXb)$, then determination of whether a payoff is earned depends on the behavior occurring within the time interval with length equal to the depth of X (in this case, the depth is 2).
- Instead of seeing a behavior at a time d later, by assigning a payoff according to a time d before, we can easily construct a mean-payoff game MG_t, which is equivalent to t.

Trade-off between size and tolerance



- we might find a reactive system rs', which has a higher degree $deg' > deg$ of environmental tolerance.
- However, the representation size of rs' is much larger than that of rs, and the computational cost for synthesis of rs' is much larger than that of rs.

There is a trade-off between size, cost and degree of environmental tolerance of synthesized reactive systems.

Implementation and Evaluation

- We implemented our method and used the LTL3BA tool[BKRS12] to convert the LTL formulae into universal coBüchi automata.
- We describe syntheses of reactive systems from a simple specification and a more practical specification at a non-trivial scale.
    - Using the minimal k that allows realization to be obtained, we compute a winning region WR_k, and deduce the optimal strategy on WR_k using an algorithm proposed in [ZP96], where we compute convergence of values on states.
    - For this algorithm, we introduce a heuristic to check the optimality of the tentative value, after the value has almost converged.

Synthesis from a simple specification

The specification for a reactive system controlling a door.

$\varphi_{env} \to \varphi_{sys1} \wedge \varphi_{sys2}$, where

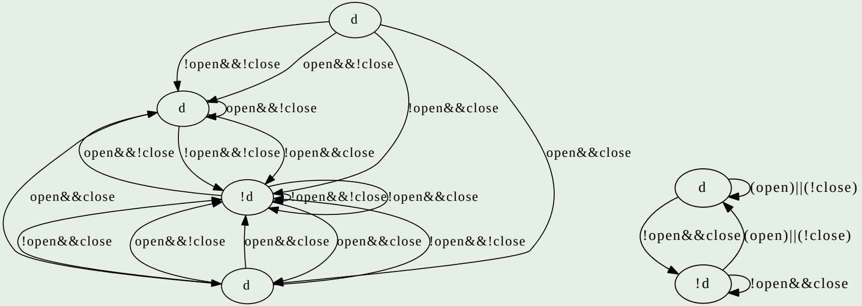
- $\varphi_{env} = G(\neg open \vee \neg close)$
- $\varphi_{sys1} = G(open \to Xd)$, and $\varphi_{sys2} = G(close \to XF\neg d)$.

Here, we approximate $F\neg d$ as $\neg d \vee X\neg d$, and obtain the following payoff term t from φ_{sys}. (we use 1 for each $c_{\varphi sysi}$.)

- $t = 1_{open \to Xd} + 1_{close \to X(\neg d \vee X\neg d))}$

Results of synthesis

A reactive system synthesized with our method (rs_1, left) and a reactive system that does not consider environmental tolerance (rs_2,right).



- Synthesis of both rs_1 and rs_2 completed within a second.
- rs_1 is environmentally more tolerant than rs_2.
  - $Tol(rs_1, t) = 3/2$. $Tol(rs_2, t) = 1$.

Evaluation

The most important difference between rs_1 and rs_2 is their behavior after the unexpected occurrence of environmental events $\{open, close\}$.
For $\tilde{i}_1 = (\{open, close\}\{\})^\omega$,

$$\begin{aligned} behave_{rs_1}(\tilde{i}_1) &= \{open, close, d\}(\{d\}\{open, close\})^\omega \\ behave_{rs_2}(\tilde{i}_1) &= (\{open, close, d\}\{d\})^\omega \end{aligned}$$

- $[\![MP(t)]\!]_{behave_{rs_1}(\tilde{i}_1)} = 2$.
  - the constraints $(open \rightarrow Xd)$ and $(close \rightarrow X(\neg d \vee X\neg d))$ are always satisfied.
- $[\![MP(t)]\!]_{behave_{rs_2}(\tilde{i}_1)} = 3/2$.
  - $(open \rightarrow Xd)$ is always satisfied, while $(close \rightarrow X(\neg d \vee X\neg d))$ is only satisfied with a frequency of $1/2$.

We confirmed that the obtained reactive system has environmental tolerance.

Synthesis from a practical specification

A specification of an m-floor elevator system

- $m+2$ environmental events, $2m+4$ system events, and $12m+8$ temporal formulae.
- The specification consists of
  - a specification for each floor $\cdots(a)$
    divided into
    - functional requirements for k-th floor $\cdots(akf)$, and
    - non-functional requirements for k-th floor $\cdots(akn)$
      (a specification for k-th floor (ak)=(akf) ∪ (akn))
  - a specification for a door of the lift $\cdots(b)$
  - a specification of the physical constraints$\cdots(c)$

A specification of an m-floor elevator system

Environmental events

$LocBtn_i (i = 1..m)$,	//Request button at ith floor is pushed.
$OpenBtn$,	//Open button in the lift is pushed.
$CloseBtn$	//Close button in the lift is pushed.

System events

$Loc_i (i = 1..m)$,	//The lift is located at ith floor.
$ReqLoc_i (i = 1..m)$,	//The lift is requested to go to ith floor.
$Open$,	//The door is open.
$Movable$,	//The lift can move.
$OpenTimedOut$,	//The time-limit that the door can open
	//has past.
$ReqOpen$	//The door is requested to open.

(a) a specification for each floor

(af) functional requirements

//If a request button is pushed, the lift eventually go there.

$$\bigwedge_{1 \leq i \leq m} G(LocBtn_i \rightarrow F\,Loc_i \wedge ReqLoc_i W (Loc_i \wedge ReqLoc_i)),$$

//If the lift reaches the requested floor, the door open.

$$\bigwedge_{1 \leq i \leq m} G(Loc_i \wedge ReqLoc_i \rightarrow Open \wedge Loc_i W\,Movable),$$

(an) non-functional requirements

//Until request button is pushed, lift is not requested to go there.

$$\bigwedge_{1 \leq i \leq m} G(Loc_i \wedge Movable \rightarrow (\neg ReqLoc_i) W\,LocBtn_i),$$

//If the lift is not requested at a floor, the door will not open there.

$$\bigwedge_{1 \leq i \leq m} G(Loc_i \wedge \neg ReqLoc_i \rightarrow \neg Open),$$

(b) a specification for a door of the lift

//The time-limit that the door open is set.
$G(Open \rightarrow F\,OpenTimedOut)$,

//If open button is pushed, the door is requested to open.
$G(OpenBtn \wedge \neg OpenTimedOut \rightarrow ReqOpen)$,

//When the time that door can open passed, the door closes.
$G(OpenTimedOut \rightarrow \neg Open)$,

//If close button is pushed, the door closes.
$G(CloseBtn \wedge \neg ReqOpen \rightarrow \neg Open)$,

//If the door is requested to open and the lift is not movable, the door opens.
$G(ReqOpen \wedge \neg Movable \rightarrow Open)$

(c) a specification of the physical constraints

//The lift is located at some floor.

$$G(\bigvee_{1\leq i\leq m} Loc_i) \land G(\bigwedge_{1\leq i\leq m} (Loc_i \rightarrow \bigwedge_{j=1..m, i\neq j} \neg Loc_j)),$$

//The lift must pass floors on the way to the destination. $(m \geq 3)$

$$G(\bigwedge_{\substack{1\leq i\leq m-2\\ 3\leq j\leq m\\ i\leq j-2}} (Loc_i \land ReqLoc_j \rightarrow \bigwedge_{i+2\leq k\leq j} (\neg Loc_k) U (\neg Loc_k \land Loc_{k-1}))),$$

$$G(\bigwedge_{\substack{1\leq i\leq m-2\\ 3\leq j\leq m\\ i\leq j-2}} (Loc_j \land ReqLoc_i \rightarrow \bigwedge_{i+2\leq k\leq j} (\neg Loc_k) U (\neg Loc_k \land Loc_{k+1}))),$$

//A relation between the door open/close and the lift movable/unmovable

$G(Open \rightarrow (\neg Movable) W (\neg Open)),$

$G(\neg Open \rightarrow Movable\, W\, Open),$

Environmental constraint and payoff term

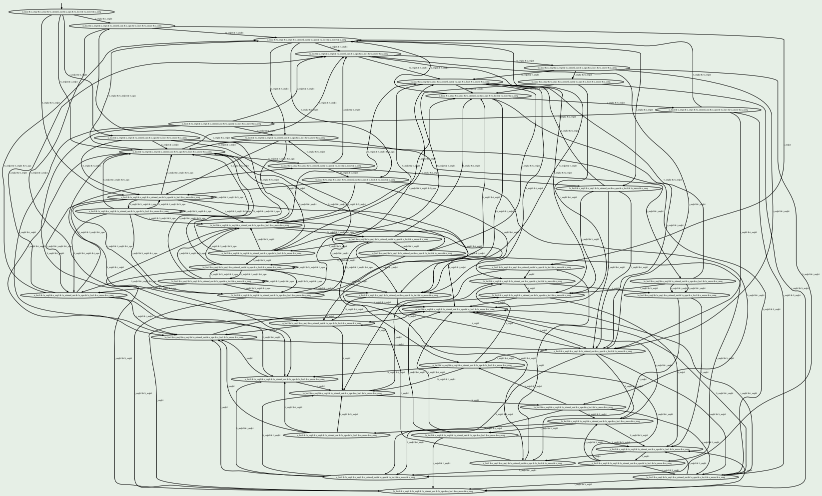
An environmental constraint

//If the floor button of each floor is pushed, then the button is released at the next time unit.
$\bigwedge_{1 \leq i \leq m} G(LocBtn_i \rightarrow X \neg LocBtn_i)$

We obtain a payoff term from the four requirements: two functional requirements of (a), (b) and (c), by using each payoff $c_i = 1$. In the payoff term we approximate the Until operator with the bounded Until operator (the bound is 2).

Results

For the case where $m = 2$, we successfully synthesized a reactive system in a reasonable period of time.



Results (cont.)

	Elevator systems	
	by our method	without tolerance
num of states	49	19
num of transitions	179	55
degree of tolerance	3.5	2.5
computation time	52.43(sec)	0.82(sec)

From these results, we confirmed that with our method we can synthesize a reactive system with environmental tolerance from the more practical specification of a non-trivial scale, although the size of the resulting reactive system is non-trivial and the computation time is larger than that for synthesis without the consideration of environmental tolerance.

Related works

- Tools for synthesizing reactive systems from specifications written in LTL
  - Lily [JB06], AcaciaPlus[BBF$^+$12, BBFR13], and Unbeast[Ehl10].
- Methods for synthesizing reactive systems with environmental tolerance [BCG$^+$14] and [Ehl11].
- A method for synthesizing optimal reactive systems from quantitative specifications [BCHJ09].
- A study on practical reactive system synthesis from LTL specifications considering unpredictable environmental behavior [DBSU15].
- A detailed survey of synthesis from specifications with environmental assumptions [BEJK14]

Conclusion

- We describe a simple quantitative definition for environmental tolerance.
- We also proposed a method for synthesizing a reactive system with environmental tolerance.
- The method was implemented, and reactive systems synthesized using our method compared to reactive systems synthesized without the consideration of environmental tolerance.
    - We confirmed that, for the unexpected behavior of an environment, the behavior of the reactive system synthesized by our method satisfied the system constraints more often than the behavior of the reactive system synthesized without the consideration of environmental tolerance.

- Aaron Bohy, Véronique Bruyère, Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin.
  Acacia+, a tool for LTL synthesis.
  In *Proceedings of the 24th international conference on Computer Aided Verification*, CAV'12, pages 652–657, Berlin, Heidelberg, 2012. Springer-Verlag.

- Aaron Bohy, Véronique Bruyère, Emmanuel Filiot, and Jean-François Raskin.
  Synthesis from LTL specifications with mean-payoff objectives.
  In *Proceedings of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'13, pages 169–184, Berlin, Heidelberg, 2013. Springer-Verlag.

- Roderick Bloem, Krishnendu Chatterjee, Karin Greimel, Thomas A. Henzinger, Georg Hofferek, Barbara Jobstmann, Bettina Könighofer, and Robert Könighofer.
  Synthesizing robust systems.
  Acta Inf., 51(3-4):193–220, June 2014.

Roderick Bloem, Krishnendu Chatterjee, Thomas A. Henzinger, and Barbara Jobstmann.
Better quality in synthesis through quantitative objectives.
In *Proceedings of the 21st International Conference on Computer Aided Verification*, CAV '09, pages 140–156, Berlin, Heidelberg, 2009. Springer-Verlag.

Roderick Bloem, Rüdiger Ehlers, Swen Jacobs, and Robert Könighofer.
How to handle assumptions in synthesis.
In Krishnendu Chatterjee, Rüdiger Ehlers, and Susmit Jha, editors, Proceedings 3rd Workshop on *Synthesis*, Vienna, Austria, July 23-24, 2014, volume 157 of *Electronic Proceedings in Theoretical Computer Science*, pages 34–50. Open Publishing Association, 2014.

Tomás Babiak, Mojmír Kretínský, Vojtech Rehák, and Jan Strejcek.
LTL to Büchi automata translation: Fast and more deterministic.
In *TACAS*, pages 95–109, 2012.

Nicolas D'Ippolito, Victor Braberman, Daniel Sykes, and Sebastian Uchitel.
Robust degradation and enhancement of robot mission behaviour in unpredictable environments.
In *Proceedings of the 1st International Workshop on Control Theory for Software Engineering*, CTSE 2015, pages 26–33, New York, NY, USA, 2015. ACM.

Rüdiger Ehlers.
Symbolic bounded synthesis.
In *Proceedings of the 22nd international conference on Computer Aided Verification*, CAV'10, pages 365–379, Berlin, Heidelberg, 2010. Springer-Verlag.

Rüdiger Ehlers.
Generalized Rabin(1) synthesis with applications to robust system synthesis.
In *Proceedings of the Third International Conference on NASA Formal Methods*, NFM'11, pages 101–115, Berlin, Heidelberg, 2011. Springer-Verlag.

Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin.
An antichain algorithm for LTL realizability.
In *Computer Aided Verification, 21st International Conference, CAV 2009*, volume 5643 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 2009.

Barbara Jobstmann and Roderick Bloem.
Optimizations for LTL synthesis.
In *Formal Methods in Computer Aided Design, 2006. FMCAD '06*, pages 117–124, 2006.

Orna Kupferman and Moshe Y. Vardi.
Safraless decision procedures.
In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '05, pages 531–542, Washington, DC, USA, 2005. IEEE Computer Society.

Amir Pnueli and Roni Rosner.
On the synthesis of a reactive module.

In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 179–190, 1989.

Roni Rosner.
Modular Synthesis of Reactive Systems.
PhD thesis, Weizmann Institute of Science, 1992.

Sven Schewe and Bernd Finkbeiner.
Bounded synthesis.
In *Automated Technology for Verification and Analysis, 5th International Symposium, ATVA 2007*, volume 4762 of *Lecture Notes in Computer Science*, pages 474–488. Springer, 2007.

Takashi Tomita, Shin Hiura, Shigeki Hagihara, and Naoki Yonezaki.
A temporal logic with mean-payoff constraints.
In *Proceedings of the 14th International Conference on Formal Engineering Methods: Formal Methods and Software Engineering*, ICFEM'12, pages 249–265, Berlin, Heidelberg, 2012. Springer-Verlag.

Uri Zwick and Mike Paterson.
The complexity of mean payoff games on graphs.

Theoretical Computer Science, 158:343–359, 1996.