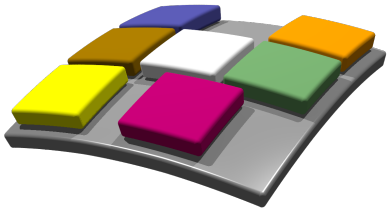

Cassting

Deliverable D3.2

Synthesis of Succinct Representations of Strategies

Wolfgang Thomas (RWTH Aachen)

Project	Cassting — Collective Adaptive Systems SynThesIs with Non-zero-sum Games		
Project id.	FP7-601148		
Workpackage	WP3	**Nature**	R (report)
Deliverable	D3.2	**Dissemination**	PU (public)
Date submitted	September 2015	**Date due**	September 2015
Version	1.0 (final)		

Synthesis of Succinct Representations of Strategies

Wolfgang Thomas (RWTH Aachen)

September 2015

1 Introduction

This report addresses an important objective of Cassting, namely to find new ways of synthesizing winning strategies in games by exploiting "structure" in various ways. To explain this, one should recall the standard approach in the game-theoretic approach to program synthesis. From a game specification (usually in the form of a game arena and a logical formula describing the winning condition for a designated player), one proceeds to a transition system and a simpler winning condition (e.g. the parity condition), from where the synthesis algorithms starts, producing an automaton (i.e., a deterministic form of transition system) that describes a winning strategy. In this approach – although successful in many contexts – two kinds of "structure" are neglected in the construction: the structure of the arena (which, for instance, may be a product of similar component structures) and the structure of the winning condition (which may be given in a compositional form when presented as a logic formula).

The objective of the Cassting task T3.2 is to devise methods which allow to exploit structure in the synthesis problem, or which produces structured forms of strategies, e.g. as "programs" rather than mere transition systems. The central aim in this endeavor is to find efficient – i.e., succinct – formats of winning strategies. Of course, a structured format as such is not enough; it should be part of an efficient synthesis process.

This problem is a fundamental one; it occurs (in different forms) in virtually all branches of computer science, not just in the Cassting context. To find general solutions seems of paramount difficulty. The results of Cassting on which we report here led to new insights and methods, and they were presented at major conferences or published in premier journals, but one has to admit that we did not at one stroke overcome the fundamental problems inherent in this study.

All teams of Cassting have been active in this field and are cited below with their results. Major parts of the work were done by the Aachen team (leader of task T3.2). We proceed as follows:

- In the next section we present work of Benedikt Brütsch of the Aachen

team that clarifies the possibilites to synthesize Boolean programs from regular specifications.

- Then we describe the results of Marcus Gelderie (Aachen), who was able to develop synthesis algorithms over structured arenas, based on a novel form of "strategy machine" (which employs that concept of Turing machine rather than a plain automaton or transition system).
- In the subsequent section, results of the Brussels team on constructing succinct strategies for safety games are described.
- In a further section, we address related work by the Mons team on simple strategies in Bach-Mazur games.
- It follows a short description of implementation oriented work by the Aalborg team.
- We give a summary and preliminary assessment in the Conclusion.

The framework of Cassting was essential for these results; they would not have been obtained without the continuous flow of discussion between the teams, as realized in the Cassting workshops and further visits between the teams.

2 Synthesis of Boolean Programs

Existing approaches to the synthesis of controllers in reactive systems typically involve the construction of transition systems such as Mealy automata. In 2011, Madhusudan proposed structured programs over a finite set of Boolean variables as a more succinct formalism to represent the controller [Mad11]. He provided an algorithm to construct such a program from a given ω-regular specification without synthesizing a transition system first. His procedure is based on two-way alternating ω-automata on finite trees that recognize the set of "correct" programs.

In the Aachen team, Benedikt Brütsch has refined this approach, obtained more elegant ways of bounding the complexity of programs, but at the same time showed an inherent problem in this methodology. The latter point was surprising; its proof involved concepts from graph theory (in particular, bounded tree width). The results were published in the journal *Information and Computation*.

In more detail, the approach can be summarized as follows: Given a regular specification, for example an LTL-formula, the task is to synthesize a Boolean program with assignments, sequentiual composition, if-then-else statements, and while-loops (over Boolean conditions). This is very different from the format of OBDD's (ordered binary decision diagrams) that code transition systems. Given a finite set B of Boolean variables, a Boolean program over B is naturally represented by a tree. It is now possible to synthesize a tree automaton that accepts *all* Boolean programs that implement a winning strategy for a regular specification (e.g., given by an LTL-formula). The idea is to

consider a Büchi automaton that accepts the "bad" behaviors, and let this automaton guess a play which will be lost by the player who performs his moves according to the given program tree. The course of a play will correspond to an infinite thread of moves through this program tree, a computation that can be compressed into a simple tree automaton.

The construction of Benedikt Brütsch produces directly a standard deterministic tree automaton, simplifying a more complex construction as supplied earlier by Madhusudan. Moreover, Madhusudan's results were generalized to a the wider class of programs with bounded delay, which may read several input symbols before producing an output symbol (or vice versa). Also a lower bound for the size of these tree automata was provided. Finally, a lower bound for the number of Boolean variables that are required for a structured program to satisfy a given LTL specification was established, almost matching the known upper bound for arbitrary (unstructured) transition systems. This result, while most pleasing in the theoretical sense and obtained by an original argument invoking results on graphs of bounded tree-width, shows a severe limitation of the formalism of Boolean programs: In general, the may be no substantial gain in efficiency. On the other hand, the Boolean program approach allows to single out programs of a desired form (e.g., to be of minimal nesting of control structures) – a possibility that is lacking in the classical framework. This feature is based on the fact that the synthesis algorithm generates in fact the whole space of winning strategies. It seems that this kind of synthesis may serve as a paradigm for future research since in applications one is usually not interested in "any" winning strategy but one which performs "optimally" in some sense.

Reference: [Brü15]

3 Strategies in Compositional Games

The doctoral dissertation of Marcus Gelderie from the Aachen team [Gel14] made substantial progress in two directions of the present task T3.2:

- It established a model of strategy respresentation which is rooted in Turing machines rather than the classical model of Mealy automata.
- It showed how to devise strategies in a compositional way if the game arena is presented as a product structure.

The first item will be explained only briefly; it is work that was done just before the Cassting project started [Gel12]. The second item, announced at the ICALP conference [Gel13], is more central to task T3.2.

The thesis was written in close cooperation between the teams of Aachen and Paris (Cachan); in fact, the two team leaders served as the reviewers of the thesis.

3.1 Turing machines as implementation of strategies

Although natural, the model of Turing machine has not been pursued in the existing literature as a concept for the representation of winning strategies in infinite games. Marcus Gelderie showed that Turing-machine-based strategy representation (by "strategy machines") – if compared to the standard setting of Mealy automata – allows for a finer classification of strategies, provides a better understanding of the structure of strategies, and – as explained later – can successfully be used in areas that are difficult to reason about with automaton strategies, such as composition of games and strategies.

The work provided a formal definition of strategy machines and adapts the classical Turing-machine complexity measures to the setting of strategies. The measures are called latency, space requirement, and size. The latency is the number of computation steps that is required to produce a next move, the space requirement states how much memory is required for this computation and the size is the number of control states of the machine. These complexity measures are not present in representations of strategies that are based on automata and thus provide a finer view on strategies than currently used models.

It was shown that strategy machines of polynomial size with a polynomial space requirement can be synthesized for Muller games. Moreover, this result is extended for Streett games, where a polynomial latency can be guaranteed in addition to these bounds. The synthesis procedure is an adaption of Zielonka's algorithm, now making use of the computational power of a Turing machine to spread costly computations across the infinite play.

3.2 Compositional strategy construction

In this work, a fundamental open problem, which is of intrinsic interest in the area of automated synthesis, is t addressed: to exploit the compositional structure of an arena to derive a compositional representation of a winning strategy. For instance, if an arena is viewed as a product of several smaller transition systems, is it possible to lift this structure to strategies in games on this arena?

The classical results depend on the representation of a winning strategy by an automaton. None of these results allows to transfer a given composition of an arena into a composition of automata in such a way that a winning strategy is implemented. Since there is no lack of methods for composing automata (for example, the cascade product), it rather seems that automata are too "coarse" a tool to capture this compositional structure.

We studied the compositional nature of winning strategies in games played on products of arenas. Products of arenas can be defined in a variety of ways. As a first step towards a compositional approach to synthesis, we restricted ourselves to two notions, parallel and synchronized product. Our notion of strategy composition relies on the Turing machine based model for strategy representation mentioned above, called a strategy machine. Using this model, we showed how winning strategies in reachability games and Büchi games can

be composed from winning strategies in games over the constituent factors of the overall product arena. We studied the complexity of such a composition: its size, its runtime and the computational complexity of finding it. This entails a study of the complexity of deciding who wins the game.

Compositionality in an arena is closely linked to a succinct representation of that arena. Likewise, composing a winning strategy from smaller winning strategies may yield a much smaller representation for that strategy. Transition systems which are obtained by "multiplying" smaller transition systems have also been studied in the literature. It is known that even the model checking problem for such systems may be of high complexity for various notions of behavioral specification.

With our model of strategy machine, we can estimate the "runtime" of a strategy and quantify and compare "dynamic" memory (the tape content) and "static memory" (the control states).

In our work we focus on two notions of product of arenas, the parallel product and the synchronized product. The games we study are played on arenas that are composed from smaller arenas using these two operators. Having defined the notion of arena composition, we used strategy machines to introduce our notion of strategy composition. Subsequently, this approach was first applied to reachability games, then to Büchi games, separately for the parallel product and the synchronized product. To this end, we first introduced two natural ways of defining a reachability condition on a composite arena, local and synchronized reachability. For the parallel product we obtained a compositionality theorem for both local and synchronized reachability. For the synchronized product we showed that deciding the game is Exptime complete. Our strategy composition based on subroutine calls and showed that as long as the component arenas cannot communicate, a composition of polynomial size and latency can be computed in polynomial time.From this we deduce that finding a general composition theorem is equivalent to showing Exptime = Pspace. This is a surprising link of the Cassting task T3.2 to hard open problems in complexity theory and also illustrates the inherent difficulty of compositional strategy construction.

Reference: [Gel13], [Gel14]

4 Synthesising Succinct Strategies in Safety Games

In this work by the Brussels team, co-authored by Gilles Geeraerts, Joël Goossens, and Amélie Stainer, a novel general technique was devised to compute, *efficiently, succinct representations* of winning strategies in safety and reachability games. This technique adapts the *antichain* framework (developed earlier by members of the Brussels team, e.g., J. F. Raskin) to the setting of games. It relies on the notion of *turn-based alternating simulation*, which is used to formalise natural relations that exist between the states of those games in many

applications. In particular, the technique applies to the realisability problem of LTL [FJR11a], to the synthesis of real-time schedulers for multiprocessor platforms [BM10], and to the determinisation of timed automata [BSJK11] – three applications where the size of the game one needs to solve is at least exponential in the size of the problem description, and where succinct strategies are particularly crucial in practice.

Let us explain this work in a little more detail. Finite, turn-based, safety games are arguably one of the most simple, yet relevant, classes of games. They are played by two players (A and B) on a finite graph (called the arena), whose set of vertices is partitioned into Player A and Player B vertices, (that we call A and B-states respectively). A play is an infinite path in this graph, and is obtained by letting the players move a token on the vertices. Initially, the token is on a designated initial vertex. At each round of the game, the player who owns the vertex marked by the token decides on which successor node to move it next. A play is winning for A if the token never touches some designated bad nodes, otherwise, it is winning for B.

Such games are a natural model to describe the interaction of a potential controller with a given environment, where the aim of the controller is to avoid the bad states that model system failures. In this framework, computing a winning strategy for the player amounts to *synthesising* a control policy that guarantees no bad state will be reached, no matter how the environment behaves. Safety games have also been used as a tool to solve other problems, as mentioned above.

One of the nice properties of safety games is that they admit *memory-less winning strategies*, i.e., if Player A has a winning strategy in the game, then he has a winning strategy that depends only on the current state (in other words, this strategy is a function from the set of A-states to the set of B-states). Memory-less strategies are often regarded as simple and straightforward to implement (remember that the winning strategy is very often the actual control policy that we want to implement in, say, an embedded controller). Yet, this belief falls short in many practical applications such as the three mentioned above because the arena is not given explicitly, and its size is *at least exponential* in the size of the original problem instance. Hence, two difficulties arise, even with memory-less strategies. First, the computation of winning strategies might request the traversal of the whole arena, which might be intractable in practice. Second, a naive implementation of a winning strategy σ by means of an exponentially large table, mapping each A-state v to its safe successor $\sigma(v)$, is not realistic.

The work reported here addresses the problem of computing *efficiently* winning strategies (for Player A) that can be *succinctly* represented. To formalise this problem, we considered $\star$-*strategies*, which are *partial functions* defined on a subset of A-states only. A $\star$-strategy can be regarded as an *abstract representation* of a family of (plain) strategies, that we call *concretisations* of the $\star$-strategies (they are all the strategies that agree with the $\star$-strategy). Then, we computed *succinct* $\star$-strategies that are defined on the smallest possible number of states, but that are still safe because all their concretisations are winning.

Unfortunately, we show that, unless P=NP, no polynomial-time algorithm exists to compute minimal $\star$-strategies, because the associated decision problem is NP-complete. This holds when the arena is given explicitly, so the difficulty is exacerbated in practice, where the arena is much larger than the problem instance.

To cope with this complexity, we considered heuristics inspired from the *antichain* line of research [DR10]. Antichain techniques rely on a *simulation partial order* on the states of the system. This simulation is exploited to prune the state space that the algorithms need to explore, and to obtain efficient data structures to store the set of states that the algorithms need to maintain. These techniques have been applied to several relevant problems, such as LTL model-checking [DR10] or multi-processor schedulability [GGL13] with remarkable performance improvements of several orders of magnitude.

While a general theory of antichain has been developed [DR10] in the setting of automata-based models (that are well-suited for verification), it has been scarcely applied in the setting of games. One notable exception is the aforementioned work of Filiot *et al.* on LTL realisability [FJR11b] where the problem of realisability is reduced to a safety game, and a simulation on the game states is exploited in an efficient algorithm tailored for those games. The heuristics are thus limited to this peculiar case, and the correctness cannot always be deduced from the notion of simulation but requires ad hoc proofs.

We advocated the use of a different notion, *turn-based alternating simulations* (tba-simulations for short). Tba-simulations allow us to develop an elegant and general theory that extends the ideas of Filiot *et al.* and that is applicable to a broad class of safety games.

It urns out that it is then easy to implement *succinctly* this strategy: the simulation relation can be directly computed on the description of the states. All these intuitions were formalised; we showed that, in general, it is sufficient to store the strategy on the maximal antichain of the reachable winning states, i.e., on a set of states that are all incomparable by the tba-simulation, and thus very compact.

Next, we presented *an efficient on-the-fly algorithm to compute such succinct $\star$-strategies*. Our algorithm incorporates several heuristics that stem directly from the definition of tba-simulation (in particular, it incorporates an optimisation that was not present in the work of Filiot et al. [FJR11b]). Finally, we devised a criterion that allows to determine when a simulation relation on a game arena is also a tba-simulation. We showed that the safety games one considers in three applications introduced above (LTL realisability, real-time feasibility and determinisation of timed automata) respect this criterion, which demonstrates the wide applicability of our approach.

Reference: [GGS14a], [GGS14b]

5 Simple Strategies for Banach-Mazur Games

In this work, the outset was given by the observation [VV06] that 'sometimes, a model of a concurrent or reactive system does not satisfy a desired linear-time temporal specification but the runs violating the specification seem to be artificial and rare'. As a naive example of this phenomenon, consider a coin flipped an infinite number of times. Classical verification will assure that the property stating "one day, we will observe at least one head" is false, since there exists a unique execution of the system violating the property. In some situations, for instance when modelling non-critical systems, one could prefer to know whether the system is *fairly correct*. Roughly speaking, a system is fairly correct against a property if the set of executions of the system violating the property is "very small"; or equivalently if the set of executions of the system satisfying the property is "very big". A first natural notion of a fairly correct system is related to probability: almost-sure correctness. A system is almost-surely correct against a property if the set of executions of the system satisfying the property has probability 1. Another interesting notion of fairly correct system is related to topology: large correctness. A system is largely correct against a property if the set of executions of the system satisfying the property is large (in the topological sense). There exists a nice characterisation of large sets by means of the *Banach–Mazur games*. In these games, the two players choose sequences of letters (finite words) rather than single letters in each of the moves. It is known that a set W is large if and only if Player 0 has a winning strategy in the related Banach–Mazur game.

Although the two notions of fairly correct systems do not coincide in general, in [VV06] it was shown that when considering ω-regular properties on finite systems, the almost-sure correctness and the large correctness coincide, for bounded Borel measures. Motivated by this very nice result, we intended to extend it to a larger class of specifications. The key ingredient to prove the previously mentioned result of [VV06] is that when considering ω-regular properties, positional strategies are sufficient in order to win the related Banach–Mazur game. For this reason, we investigated simple strategies in Banach–Mazur games.

We first compared various notions of simple strategies on finite graphs (including bounded and move-counting strategies), and their relations with the sets of probability 1. Given a set W , the existence of a bounded (resp. move-counting) winning strategy in the related Banach–Mazur game implies that W is a set of probability 1. However there exist sets W of probability 1 for which there is no bounded and no move-counting winning strategy in the related Banach–Mazur game. Therefore, we introduced a generalisation of the classical Banach–Mazur game and in particular, a probabilistic version whose goal is to characterise sets of probability 1 (as classical Banach–Mazur games characterise large sets). We obtained the desired characterisation in the case of countable intersections of open sets. This is the main contribution of this work. As a byproduct of the latter, we got a determinacy result for our probabilistic version of the Banach–Mazur game for countable intersections of open sets. We finally extended this probabilistic version to Banach–Mazur games on topolog-

ical spaces.

The first publication of this work was the paper "Simple strategies for Banach–Mazur games and fairly correct system" published in the proceedings of GandALF 2013. In the extended version, we added a comparison of move/length-counting strategies by showing that one is a uniform version of the other. We investigated the minimal necessary memory to keep the full power of winning strategies and we extended our notion of generalised Banach–Mazur games (originally only defined on finite graphs) to topological spaces.

References: [BM13], [BHM15] (in press, with additional author Axel Haddad).

6 Uppaal Stratego

In this work, co-authored by Alexandre David, Peter G. Jensen, Kim G. Larsen, Marius Mikuvcionis, and Jakob H. Taankvist, Uppaal Stratego was a novel tool which facilitates generation, optimization, comparison as well as consequence and performance exploration of strategies for stochastic priced timed games in a user-friendly manner. The tool allows for efficient and flexible "strategy-space" exploration before adaptation in a final implementation by maintaining strategies as first class objects in the model-checking query language.

Let us start with recalling relevant background. Model checking may be used to verify that a proposed controller prevents an environment from causing dangerous situations while, at the same time, operating in a desirable manner. This approach has been successfully pursued in the setting of systems modeled as finite-state automata, timed automata, and probabilistic automata of various types with tools such as nuSMV, FDR, Uppaal, and PRISM as prime examples of model checking tools supporting the above mentioned formalisms. Most recently the simulation-based method of statistical model checking has been introduced in Uppaal SMC, allowing for highly scaleable analysis of fully stochastic Sliced Timed Automata with respect to a wide range of performance properties. For instance, expected waiting-time and cost, and time-bounded and cost reachability probabilities, may be estimated (and tested) with an arbitrary precision and high degree of confidence. Combined with the symbolic model checking of Uppaal this enables an adequate analysis of mixed critical systems, where certain (safety) properties must hold with absolute certainty, whereas for other quantitative (performance) properties a reasonably good estimation may suffice.

Rather than verifying a proposed controller, synthesis—when possible—allows an algorithmic construction of a controller which is guaranteed to ensure that the resulting systems will satisfy the desired correctness properties. The extension of controller synthesis to timed and hybrid games started in the 90s with seminal work of Pnueli et al. on controller synthesis for timed games, where the synthesis problem was proven decidable by a symbolic dynamic programming technique. In Uppaal Tiga, an efficient on-the-fly algorithm for synthesis of

reachability and safety objectives for timed games has been implemented, with a number of successful industrial applications having been made including zone-based climate control for pig-stables and controllers for hydraulic pumps with 60% improvement in energy-consumption compared with industrial practice at the time.

However, once a strategy has been synthesized for a given objective no further analysis has been supported so far. In particular it has not been possible to make a deeper examination of a synthesized strategy in terms of other additional properties that may or may not hold under the strategy. Neither has it been possible to optimize a synthesized non-deterministic safety strategy with respect to desired performance measures. Both of these issues have been addressed by the authors, and in the work reported here tool Uppaal Stratego is presented which combines these techniques to generate, optimize, compare and explore consequences and performance of strategies synthesized for stochastic priced timed games in a user-friendly manner. In particular, the tool allows for efficient and flexible "strategy-space" exploration before adaptation in a final implementation.

Uppaal Stratego integrates Uppaal and the two branches Uppaal SMC (statistical model checking), Uppaal Tiga (synthesis for timed games) and a method proposed in [DJL+14] (synthesis of near optimal schedulers) into one tool suite. Uppaal Stratego comes with an extended query language where strategies are first class objects that may be constructed, compared, optimized and used when performing (statistical) model checking of a game under the constraints of a given synthesized strategy.

The typical scenario is a stochastic priced timed game (SPTG) which models, in a simple example, one person reading the newspaper. The model reflects the reading of the four sections in the preferred order (here comics, sport, local and economy) for the preferred amount of time. In certain locations the person is waiting for the next section to become available; here four Boolean variables are used to ensure mutex on the reading of a section. In other locations, the person is reading the particular section for a duration given by a uniform distribution on the given interval, e.g. for our person's reading of sport. A stopwatch WTime is included and only runs in the waiting locations, thus effectively measuring the accumulated time when the person is waiting to read.

Given a complete model with several persons constantly competing for sections accessible only by one player, we showed how to synthesoize strategies for several multi-objectives, e.g. strategies ensuring that all persons have completed reading within 100 minutes, and then minimize the expected waiting time for our preferred person.

Reference: [DJL+15]

7 Conclusion

The report shows that substantial progress and interesting results have been obtained on the Cassting Task T 3.2, and it show that all Cassting teams have obtained non-trival contributions. Most of this work was published in high-level venues (conferences, journals).

Although the results mentioned here would not have been brought to life without Cassting, due to a lively scientific exchange during the Cassting workshops, more integration of the different threads of research would have been desirable. The perspective of reaching a uniform, powerful, and at the same time efficient approach for the various synthesis problems mentioned here turned too ambitious. However, a lot of progress was made in joining different approaches. This shows that the joint efforts of the Cassting teams regarding task T3.2 were successful to a good extent.

List of Cassting Publications

[BHM15] Thomas Brihaye, Axel Haddad, and Quentin Menet. Simple strategies for Banach-Mazur games and sets of probability 1. *Information and Computation*, 2015. To appear.

[BM13] Thomas Brihaye and Quentin Menet. Simple strategies for Banach-Mazur games and fairly correct systems. In Gabriele Puppis and Tiziano Villa, editors, *Proceedings of the 4th International Symposium on Games, Automata, Logics, and Formal Verification (GandALF'13)*, volume 119 of *Electronic Proceedings in Theoretical Computer Science*, pages 21–34, Borca di Cadore, Italy, August 2013. URL: `http://www.cassting-project.eu/wp-content/uploads/BM-gandalf13.pdf`, `doi:10.4204/EPTCS.119.5`.

[Brü15] Benedikt Brütsch. Synthesizing structured reactive programs via deterministic tree automata. *Information and Computation*, 242(0):108–127, 2015. `doi:10.1016/j.ic.2015.03.013`.

[DJL+15] Alexandre David, Peter G. Jensen, Kim G. Larsen, Marius Mikucvcionis, and Jakob H. Taankvist. Uppaal stratego. In Christel Baier and Cesare Tinelli, editors, *Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'15)*, volume 9035 of *Lecture Notes in Computer Science*, pages 206–211, London, UK, April 2015. Springer. URL: `http://www.cassting-project.eu/wp-content/uploads/DJLMT-tacas15.pdf`, `doi:10.1007/978-3-662-46681-0_16`.

[Gel13] Marcus Gelderie. Strategy composition in compositional games. In Fedor V. Fomin, Rusins Freivalds, Marta Kwiatkowska, and David Peleg, editors, *Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP'13) – Part II*, volume 7966 of *Lecture Notes in Computer Science*, pages 263–274, Riga, Latvia, July 2013. Springer. URL: `http://www.cassting-project.eu/wp-content/uploads/Gel13.pdf`, `doi:10.1007/978-3-642-39212-2_25`.

[Gel14] Marcus Gelderie. *Strategy Machines: Representation and Complexity of Strategies in Infinite games*. PhD thesis, RWTH Aachen University, Aachen, 2014. URL: `http://publications.rwth-aachen.de/record/229827`.

[GGS14a] Gilles Geeraerts, Joël Goossens, and Amélie Stainer. Synthesising succinct strategies in safety games. In Joël Ouaknine, Igor Potapov, and James Worrell, editors, *Proceedings of the 8th Workshop on Reachability Problems in Computational Models (RP'14)*, volume 8762 of *Lecture Notes in Computer Science*, pages 98–111, Oxford, UK, September 2014. Springer. URL: `http://`

www.cassting-project.eu/wp-content/uploads/GGS-rp14.pdf, doi:10.1007/978-3-319-11439-2_8.

[GGS14b] Gilles Geeraerts, Joël Goossens, and Amélie Stainer. Synthesising succinct strategies in safety games. *CoRR*, abs/1404.6228, 2014. URL: http://arxiv.org/abs/1404.6228.

References

[BM10] Vincenzo Bonifaci and Alberto Marchetti-Spaccamela. Feasibility analysis of sporadic real-time multiprocessor task systems. In Mark de Berg and Ulrich Meyer, editors, *Algorithms - ESA 2010, 18th Annual European Symposium, Liverpool, UK, September 6-8, 2010. Proceedings, Part II*, volume 6347 of *Lecture Notes in Computer Science*, pages 230–241. Springer, 2010. URL: http://dx.doi.org/10.1007/978-3-642-15781-3_20, doi:10.1007/978-3-642-15781-3_20.

[BSJK11] Nathalie Bertrand, Amélie Stainer, Thierry Jéron, and Moez Krichen. A game approach to determinize timed automata. In Martin Hofmann, editor, *Foundations of Software Science and Computational Structures - 14th International Conference, FOSSACS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*, volume 6604 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 2011. URL: http://dx.doi.org/10.1007/978-3-642-19805-2_17, doi:10.1007/978-3-642-19805-2_17.

[DJL+14] Alexandre David, Peter Gjøl Jensen, Kim Guldstrand Larsen, Axel Legay, Didier Lime, Mathias Grund Sørensen, and Jakob Haahr Taankvist. On time with minimal expected cost! In Franck Cassez and Jean-François Raskin, editors, *Automated Technology for Verification and Analysis - 12th International Symposium, ATVA 2014, Sydney, NSW, Australia, November 3-7, 2014, Proceedings*, volume 8837 of *Lecture Notes in Computer Science*, pages 129–145. Springer, 2014. doi:10.1007/978-3-319-11936-6_10.

[DR10] Laurent Doyen and Jean-François Raskin. Antichain algorithms for finite automata. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6015 of *Lecture Notes in Computer Science*, pages 2–22. Springer, 2010. doi:10.1007/978-3-642-12002-2_2.

[FJR11a] Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin. Antichains and compositional algorithms for LTL synthesis. *Formal Methods in System Design*, 39(3):261–296, 2011. URL: `http://dx.doi.org/10.1007/s10703-011-0115-3`, `doi:10.1007/s10703-011-0115-3`.

[FJR11b] Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin. Antichains and compositional algorithms for LTL synthesis. *Formal Methods in System Design*, 39(3):261–296, 2011. `doi:10.1007/s10703-011-0115-3`.

[Gel12] Marcus Gelderie. Strategy machines and their complexity. In Branislav Rovan, Vladimiro Sassone, and Peter Widmayer, editors, *Mathematical Foundations of Computer Science 2012 - 37th International Symposium, MFCS 2012, Bratislava, Slovakia, August 27-31, 2012. Proceedings*, volume 7464 of *Lecture Notes in Computer Science*, pages 431–442. Springer, 2012. `doi:10.1007/978-3-642-32589-2_39`.

[GGL13] Gilles Geeraerts, Joël Goossens, and Markus Lindström. Multiprocessor schedulability of arbitrary-deadline sporadic tasks: complexity and antichain algorithm. *Real-Time Systems*, 49(2):171–218, 2013. `doi:10.1007/s11241-012-9172-y`.

[Mad11] P. Madhusudan. Synthesizing reactive programs. In Marc Bezem, editor, *Computer Science Logic, 25th International Workshop / 20th Annual Conference of the EACSL, CSL 2011, September 12-15, 2011, Bergen, Norway, Proceedings*, volume 12 of *LIPIcs*, pages 428–442. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011. `doi:10.4230/LIPIcs.CSL.2011.428`.

[VV06] Daniele Varacca and Hagen Völzer. Temporal logics and model checking for fairly correct systems. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*, pages 389–398. IEEE Computer Society, 2006. `doi:10.1109/LICS.2006.49`.