Synthesis from Weighted Specifications with Partial Domains over Finite Words

Sarah Winter
joint work with Emmanuel Filiot and Christof Löding

Université libre de Bruxelles, Belgium

December, 2020
FSTTCS Conference

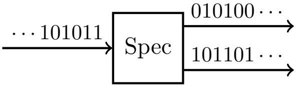
Synthesis

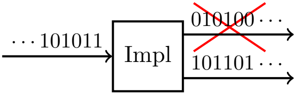
synthesize

Specification - - - - - - - - - → **Implementation**

one input is in relation
with several outputs

selects unique output
for each input

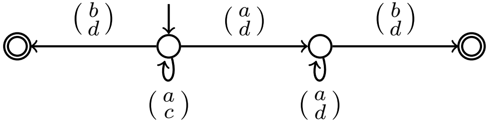




Boolean Specifications

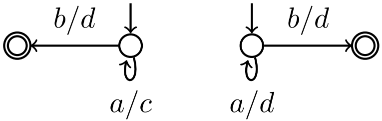
A (Boolean) **specification** is given by a synchronous deterministic automaton.

Example.



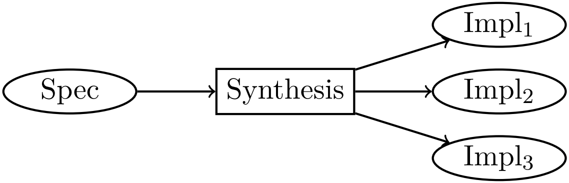
An **implementation** is given by a synchronous sequential transducer.

Example.

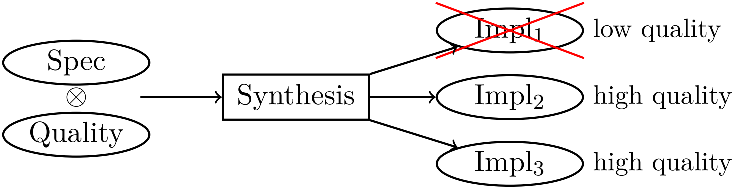


Which one is better?

Quality in Synthesis



A **quality measure** is a function $Q\colon (\Sigma \times \Gamma)^* \to \mathbb{Q}$.



How to Define High Quality?

- All executions have a lower bounded quality:

$$\forall i_1 o_1 i_2 o_2 \cdots \in [\mathrm{Impl}]\colon Q\left(\begin{smallmatrix} i_1 & i_2 & \cdots \\ o_1 & o_2 & \cdots \end{smallmatrix}\right) \geq c$$

- All executions are quality optimal:

$$\forall i_1 o_1 i_2 o_2 \cdots \in [\mathrm{Impl}]\colon Q\left(\begin{smallmatrix} i_1 & i_2 & \cdots \\ o_1 & o_2 & \cdots \end{smallmatrix}\right) = \sup_{o_1' o_2' \cdots} Q\begin{pmatrix} i_1 & i_2 & \cdots \\ o_1' & o_2' & \cdots \end{pmatrix}$$

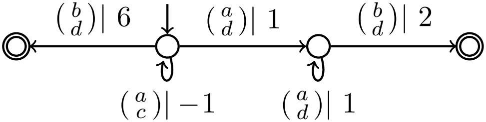
- All executions are almost quality optimal:

$$\forall i_1 o_1 i_2 o_2 \cdots \in [\mathrm{Impl}]\colon \sup_{o_1' o_2' \cdots} Q\begin{pmatrix} i_1 & i_2 & \cdots \\ o_1' & o_2' & \cdots \end{pmatrix} - Q\left(\begin{smallmatrix} i_1 & i_2 & \cdots \\ o_1 & o_2 & \cdots \end{smallmatrix}\right) \leq c$$

Weighted Specifications (:= Boolean Spec $\otimes$ Quality)

A **weighted specification** is a function $\mathsf{val}\colon (\Sigma \times \Gamma)^* \to \mathbb{Q} \cup -\infty$ given by a synchronous deterministic weighted automaton.

Example.



The value val of a pair depends on the used **payoff function**.

Example. $\mathsf{Sum}(\binom{a\ a\ b}{c\ d\ d}) = -1 + 1 + 2 = 2$, $\quad \mathsf{Sum}(aab \otimes cdd) = 2$

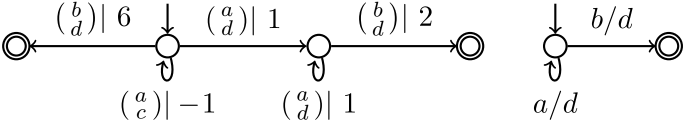
The specification **domain** is $\{u \mid \mathsf{val}(u \otimes v) \in \mathbb{Q}\}$. An input is **valid** if it is from the domain.

Example. Specification domain $= a^*b$

The **threshold synthesis problem** asks, given $c \in \mathbb{Q}$, and $\triangleright \in \{>, \geq\}$, that the implem. f satisfies for all valid inputs u:

$$\mathsf{val}(u \otimes f(u)) \triangleright c$$

Example. Sum-**specification and Implementation**



Implementation ensures value of at least 3 for all pairs.
$\mathsf{Sum}(b \otimes d) = 6, \quad \mathsf{Sum}(a^i b \otimes d^{i+1}) = i \cdot 1 + 2$

Note: Implementation can do anything on invalid inputs.

Threshold Synthesis

Spec / Problem	Sum-automata	Avg-automata	Dsum-automata
strict threshold	$\mathrm{NP} \cap \mathrm{coNP}$	$\mathrm{NP} \cap \mathrm{coNP}$	NP
non-strict threshold	$\mathrm{NP} \cap \mathrm{coNP}$	$\mathrm{NP} \cap \mathrm{coNP}$	$\mathrm{NP} \cap \mathrm{coNP}$

How to solve? See it as a game problem.

We introduce a new type of game.

Critical prefix games

Critical Prefix Games

Tailored to handle

- finite inputs
- partial specification domains

A **critical prefix game** is an **infinite-duration** two-player turn-based **weighted game** with **critical vertices**.

- When a play is in a critical vertex, quantitative constraints on the prefix are checked,
  - if fulfilled, the play continues, otherwise Adam wins.
- Nothing checked for non-critical vertices.

Critical Prefix Games and Threshold Synthesis

Threshold synthesis reduces to critical prefix games with threshold conditions.

These games are decidable for sum, average, and discounted-sum payoffs.

- Sum and average critical prefix games reduce to mean-payoff games.
- Discounted-sum critical prefix games reduce to discounted-sum games.

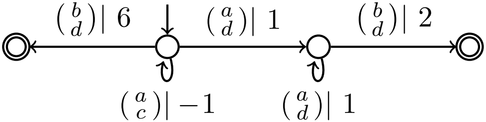
Best-value Synthesis

The **best-value synthesis problem** asks that the implementation f satisfies for all valid inputs u:

$$\mathsf{val}(u \otimes f(u)) = \mathsf{bestVal}(u) := \sup_v \mathsf{val}(u \otimes v),$$

that is, the maximal value achievable for input u.

Example. Sum-**specification**



$$\begin{aligned} \mathsf{bestSum}(b) &= 6 \\ \mathsf{bestSum}(ab) &= 5 \\ \mathsf{bestSum}(aab) &= 4 \\ \mathsf{bestSum}(aaab) &= 5 \\ \mathsf{bestSum}(aaaab) &= 6 \end{aligned}$$

No best-value implementation exists.

Best-value Synthesis

Problem \ Spec	Sum-automata	Avg-automata	Dsum-automata
best-value	PTIME [AKL10]	PTIME [AKL10]	NP ∩ coNP

Proof Techniques

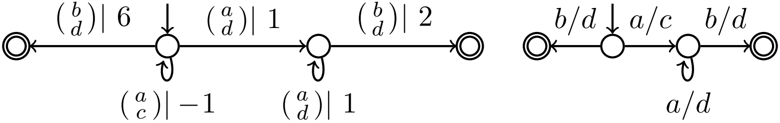
- Sum: reduces to determinization by pruning of Sum-automata
- Avg: reduces to Sum
- Dsum: reduces to a discounted-sum game

Approximate Synthesis

The **approximate synthesis problem** asks, given $c \in \mathbb{Q}$, and $\triangleleft \in \{<, \leq\}$, that the implem. f satisfies for all valid inputs u:

$$\mathsf{bestVal}(u) - \mathsf{val}(u \otimes f(u)) \triangleleft c$$

Example. Sum-**specification and Implementation**



Implementation ensures value of at most 2 less the best value.

$$\begin{array}{ll} \mathsf{Sum}(b \otimes d) = 6 & \mathsf{bestSum}(b) = 6 \\ \mathsf{Sum}(ab \otimes cd) = 5 & \mathsf{bestSum}(ab) = 5 \\ \mathsf{Sum}(a^i b \otimes c^i d) = i & \mathsf{bestSum}(a^i b) = i + 2, \quad \text{for } i \geq 2 \end{array}$$

Approximate Synthesis

Problem \ Spec	Sum-automata	Avg-automata	Dsum-automata
strict approximate	EXPTIME-c [FJL+17]	decidable EXPTIME-hard	NEXPTIME for discount $1/n$
non-strict approximate	EXPTIME-c [FJL+17]	decidable EXPTIME-hard	EXPTIME for discount $1/n$

Proof Techniques

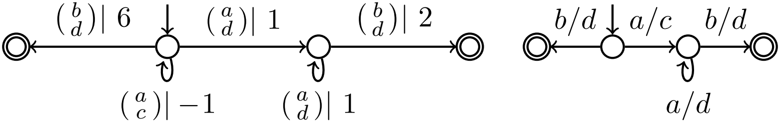
- Sum: reduces to regret determinization of Sum-automata
- Dsum: open in general, for integer discounts reduces to discounted-sum games
- Avg: best-value synthesis reduces to Sum, no longer the case for approximate synthesis

Approximate Synthesis with Average Payoff

$$\mathsf{bestAvg}(u) - \mathsf{Avg}(u \otimes v) \lhd c \quad \Leftrightarrow \quad \frac{\mathsf{bestSum}(u)}{n} - \frac{\mathsf{Sum}(u \otimes v)}{n} \lhd c$$
$$\Leftrightarrow \quad \mathsf{bestSum}(u) - \mathsf{Sum}(u \otimes v) \lhd c \cdot n$$

$$\mathsf{bestAvg}(u) - \mathsf{Avg}(u \otimes v) = 0 \quad \Leftrightarrow \quad \mathsf{bestSum}(u) - \mathsf{Sum}(u \otimes v) = 0$$

Example. Specification and Implementation



$$\mathsf{Sum}(a^i b \otimes c^i d) = i \qquad \mathsf{bestSum}(a^i b) = i + 2, \quad \text{for } i \geq 2$$
$$\mathsf{Avg}(a^i b \otimes c^i d) = \tfrac{i}{2i+2} \qquad \mathsf{bestAvg}(a^i b) = \tfrac{i+2}{2i+2}, \quad \text{for } i \geq 2$$

Reduces to (a special type of) **critical prefix games with imperfect information** which reduce to imperfect information games with fixed initial credit.

Results

Problem \ Spec	Sum-automata	Avg-automata	Dsum-automata
strict threshold	NP ∩ coNP	NP ∩ coNP	NP
non-strict threshold	NP ∩ coNP	NP ∩ coNP	NP ∩ coNP
best-value	PTIME [AKL10]	PTIME [AKL10]	NP ∩ coNP
strict approximate	EXPTIME-c [FJL+17]	decidable EXPTIME-hard	NEXPTIME for discount $1/n$
non-strict approximate	EXPTIME-c [FJL+17]	decidable EXPTIME-hard	EXPTIME for discount $1/n$

- Benjamin Aminof, Orna Kupferman, and Robby Lampert.
  Reasoning about online algorithms with weighted automata.
  ACM Trans. Algorithms, 6(2):28:1–28:36, 2010.
- Emmanuel Filiot, Ismaël Jecker, Nathan Lhote, Guillermo A. Pérez, and Jean-François Raskin.
  On delay and regret determinization of max-plus automata.
  In *LICS*, pages 1–12. IEEE Computer Society, 2017.