Retrieval of Energy Feasible Paths using Progress Measures

Automata and Applications
Supervisor: Philipp Schlehuber-Caissier
rania.saadi@epita.fr

July 2nd, 2024

In The Last Last Episode...

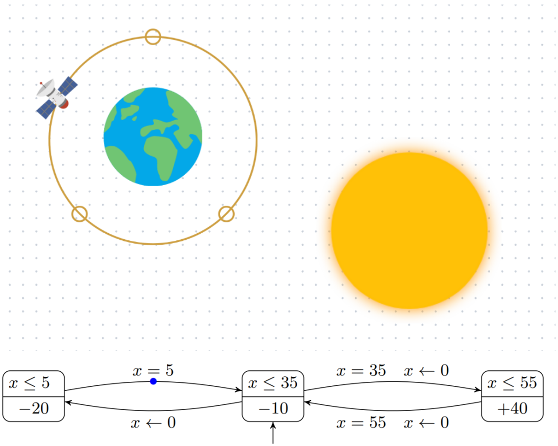


Figure 1: Modeling energy Büchi problems, automaton from "Dziadek, S., Fahrenberg, U., & Schlehuber-Caissier, P. (22 Dec 2022). Energy Büchi Problems. EPITA Research Laboratory (LRE), Paris, France.

What About Multiple Accepting Transitions?

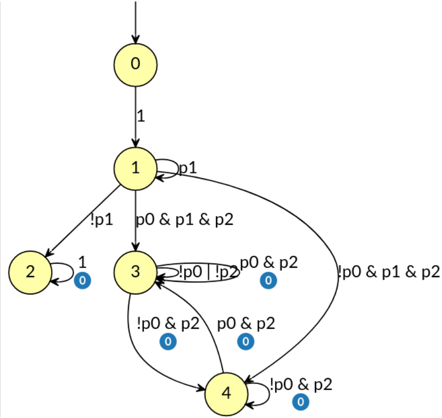


Figure 2: Automaton with Multiple Accepting Transitions - G(1U(p0 R p2)) | F!Xp1

Parity Games

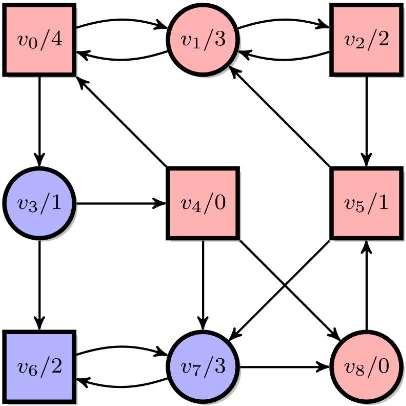


Figure 3: A Parity game (min even) form Figure 3.7 from "Zimmermann, M., Klein, F., Weinert, A. (2016). Infinite Games Lecture Notes, Summer Term. Saarland University, Germany.

Progress Measure Operators

$$s \oplus c = \begin{cases} (s_1, \ldots, s_{c-1}, 0, \ldots, 0) & \text{if } c \text{ is even,} \\ (s_1, \ldots, s_{c-2}, s_{c-1} + 1, 0, \ldots, 0) & \text{if } c \text{ is odd and} \\ & \quad c_0 = \max\{c^* \leq c \,|\, c^* \text{ odd and} \\ & \quad s_{c^*} < n_{c^*}\} \text{ is defined,} \\ T & \text{otherwise.} \end{cases}$$

$$Lift_v(\mathcal{P})(u) = \begin{cases} \mathcal{P}(u) & \text{if } u \neq v, \\ \max\{\mathcal{P}(v), \min\{\mathcal{P}(v_0) \oplus \Omega(v) \,|\, (v, v_0) \in E\}\} & \text{if } u = v \text{ and } u \in V_0, \\ \max\{\mathcal{P}(v), \max\{\mathcal{P}(v_0) \oplus \Omega(v) \,|\, (v, v_0) \in E\}\} & \text{if } u = v \text{ and } u \in V_1. \end{cases}$$

Stablization

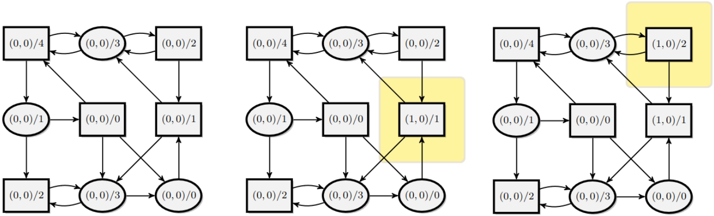


Figure 4: First Steps - Progress measure algorithm on the game from Figure 3.12 from "Zimmermann, M., Klein, F., Weinert, A. (2016). Infinite Games Lecture Notes, Summer Term. Saarland University, Germany."

Stablization

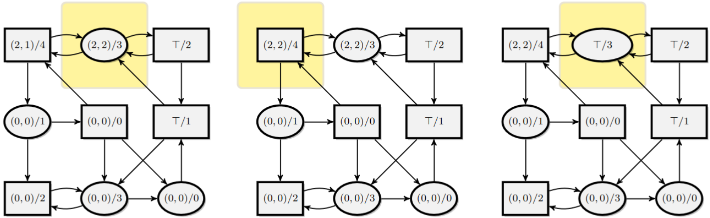


Figure 5: More iterations - Progress measure algorithm on the game from Figure 3.12 from "Zimmermann, M., Klein, F., Weinert, A. (2016). Infinite Games Lecture Notes, Summer Term. Saarland University, Germany."

Stablization

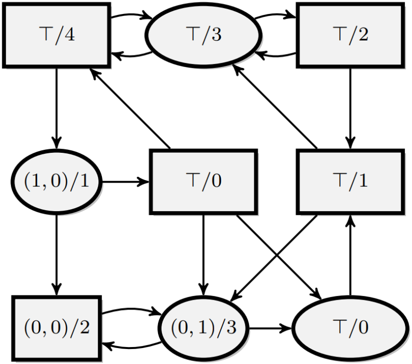
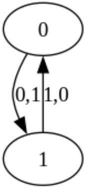


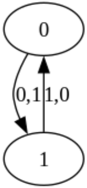
Figure 6: Stabilizing - Progress measure algorithm on the game from Figure 3.12 from "Zimmermann, M., Klein, F., Weinert, A. (2016). Infinite Games Lecture Notes, Summer Term. Saarland University, Germany."

Our Extended Score Sheets



$$\text{sc} = (\text{ic}, \text{cnt}_b) = \text{sc}' \odot B_w(w, \text{acc}) =$$
$$\begin{cases} (\text{ic}' \ominus_0 w, \text{cnt}'_b \oplus_M \text{acc}) & \text{if ic}' \ominus_0 w \leq b \\ \bot & \text{else} \end{cases}$$

Our Extended Score Sheets



$$\mathrm{sc} = (\mathrm{ic},\ \mathrm{cnt}_b) = \mathrm{sc}' \odot B_w(w, \mathrm{acc}) =$$
$$\begin{cases} (\mathrm{ic}' \ominus_0 w, \mathrm{cnt}_b' \oplus_M \mathrm{acc}) & \text{if } \mathrm{ic}' \ominus_0 w \leq b \\ \perp & \text{else} \end{cases}$$

$$\mathrm{M} = (\mathrm{b} + 1)(\mathrm{Nacc} + 1)$$

Our Extended Score Sheets

$$\mathrm{Lift}_v(\rho)(u) = \begin{cases} \rho(v) & \text{if } u \neq v \\ \bigotimes\left(\{\rho(v') \odot B_w(w, \mathrm{acc}) \mid (v, w, \mathrm{acc}, v') \in T\}\right) & \text{otherwise} \end{cases}$$

Lattices

A lattice is an abstract structure studied in the mathematical subdisciplines of order theory and abstract algebra. It consists of a partially ordered set in which every pair of elements has a unique supremum (also called a least upper bound or join) and a unique infimum (also called a greatest lower bound or meet).

- https://en.wikipedia.org/wiki/Lattice_(order)

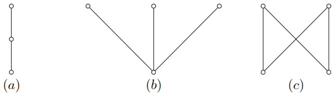


Figure 7: (a) a lattice, (b) a semilattice, (c) not a lattice

The Algorithm: PM Update

Algorithm PM Update

$PM \leftarrow [(0,0), \ldots, (0,0)]$
$change \leftarrow$ **true**
$acceptanceBound \leftarrow (wup + 1)(N_{acc} + 1)$
while $change$ **do**
    $change \leftarrow$ **false**
    for each v **in** V **do**
        for each (u, w, acc, v) **in** G **do**
            for each pmv **in** $PM[v]$ **do**
                $currentPM \leftarrow PM[u]$
                $newAcc \leftarrow \min(acceptanceBound\ ,\ pmv.acc + acc)$
                $newIc \leftarrow \max(0\ ,\ pmv.ic - w)$
                if $newIc > wup$ **then**
                    $newIc \leftarrow +\infty$
                    $newAcc \leftarrow -\infty$
                end if
                $PM[u] \leftarrow (newIc, newAcc)$
                $change \leftarrow change$ **or** $PM[u] <> currentPM$
            end for
        end for
    end for
end while

The Algorithm: PM Update (not quite)

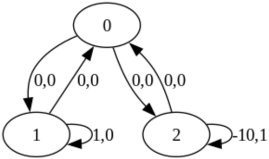


Figure 8: Counter example for the initial extended score sheet solution

ParetoJoin

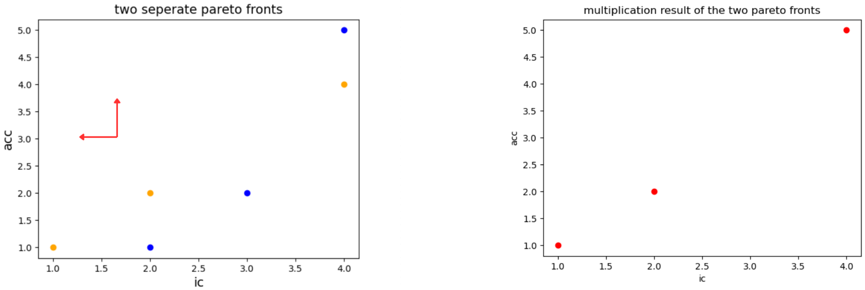


Figure 9: The pareto front join

The Algorithm: PM Update (for real)

Algorithm PM Update

```
PM ← [[(0,0)], ..., [(0,0)]]
change ← true
acceptanceBound ← (wup + 1)(N_acc + 1)
while change do
    change ← false
    for each v in V do
        for each (u, w, acc, v) in G do
            for each pmv in PM[v] do
                newAcc ← min(acceptanceBound, pmv.acc + acc)
                newIc ← max(0, pmv.ic − w)
                if newIc > wup then
                    newIc ← +∞
                    newAcc ← −∞
                end if
                change ← ParetoJoin(PM, u, newIc, newAcc)
            end for
        end for
    end for
end while
```

The Algorithm

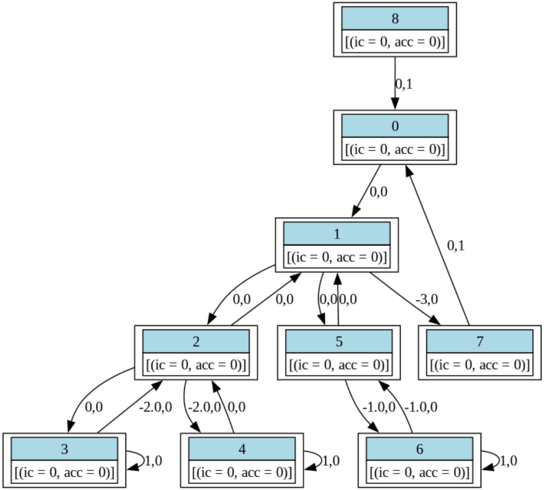


Figure 10: An automaton with score sheets - Iteration number 0

The Algorithm

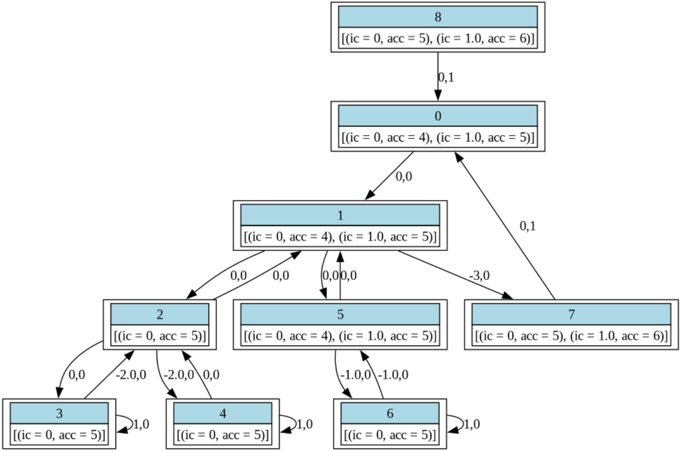


Figure 11: An automaton with score sheets - Iteration number 70

The Algorithm

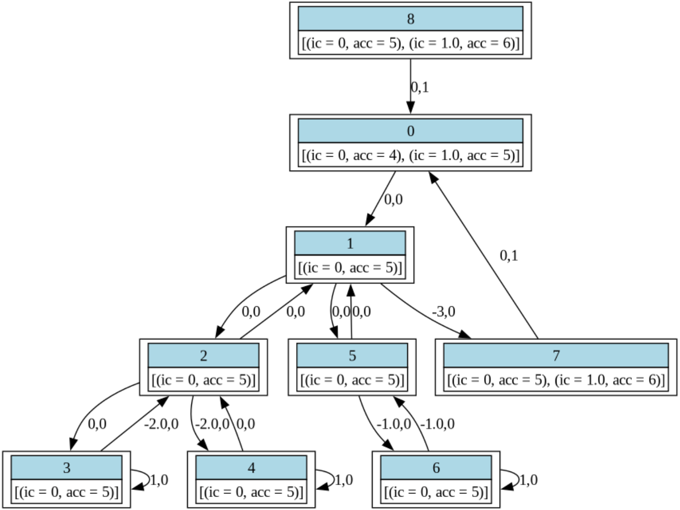


Figure 12: An automaton with score sheets - Iteration number 71

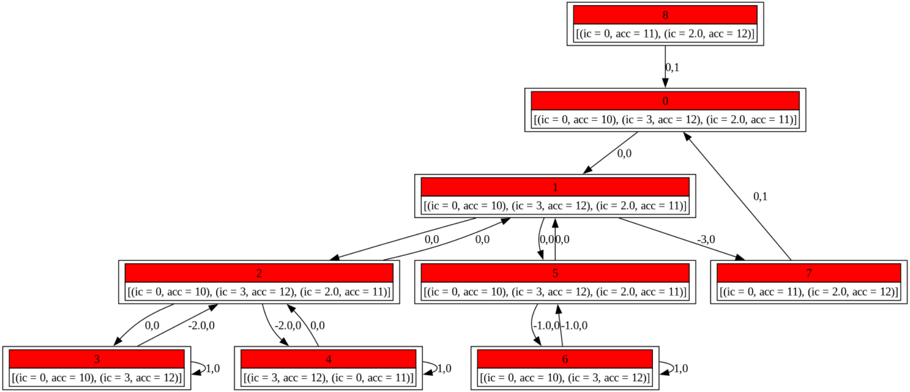


Figure 13: An automaton with score sheets - Iteration number 147

The Algorithm

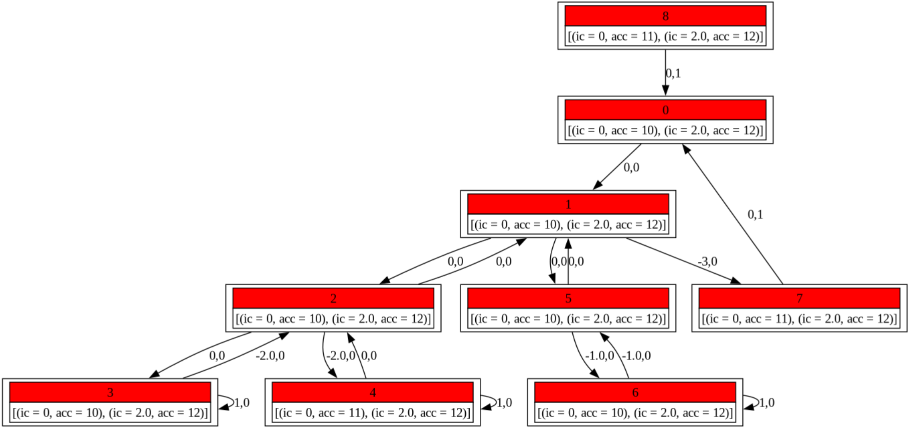


Figure 14: An automaton with score sheets - Iteration number 149

Conclusion

- Extension to parity condition
- Extension to games (büchi and parity)
- Potential for parallelization
- Improvements on the upper bound

```
           ___
     _..--"\  `|`""--.._
  .-'       \  |        `'-.
 /           \_|___...----'`\
|__,,..--""``(_)--..__      |
'\     _.--'`.I._     ''--..'
  `''"`,#JGS/_|_\###,---'`
    ,#'  _.:`___`:-._ '#,
   #'  ,~'-;(oIo);-'~, '#
   #   `~-(  |    )=~`  #
   #       | |_  |      #
   #       ; ._. ;      #
   #  _..-;|\ - /|;-._  #
   #-'   /_ \\_// _\  '-#
 /`#    ; /__\-'__\;    #`\
;  #\.--|  |O  O   |'-./#  ;
|__#/   \ _;O__O___/   \#__|
 | #\    [I_[_]__I]    /# |
 \_(#   /  |O  O   \   #)_/
       /   |        \
      /    |         \
     /    /\          \
    /     | `\         ;
   ;      \   '.       |
    \-._.__\     \_..-'/
     '.\  \-.._.-/  /'`
        \_.\    /._/
         \_.;  ;._/
       .-'-./  \.-'-.
      (___.'    '.___)
```

Thank you for your attention!

References

- Dziadek, S., Fahrenberg, U., Schlehuber-Caissier, P. (2024). Regular Energy Problems. EPITA Research Laboratory (LRE), France. Form. Asp. Comput., 1(1), March 2024.
- Chatterjee, K., Doyen, L. (2012). Energy Parity Games. IST Austria (Institute of Science and Technology Austria), Austria; LSV, ENS Cachan CNRS, France.
- Zimmermann, M., Klein, F., Weinert, A. (2016). Infinite Games Lecture Notes, Summer Term. Saarland University, Germany.
- LRDE. (n.d.). Spot - LRDE - epita. Retrieved from https://www.lrde.epita.fr/wiki/Spot.
- Nation, J. B. (n.d.). Notes on Lattice Theory. University of Hawaii.