A HASKELL encoding for reconfigurable timed systems

Antonio Iglesias, Alexandre Madeira & Manuel Martins

University of Aveiro

ReacTS25, Toledo, November 11th 2025

OUTLINE

Motivation and Objectives

Motivation

- The continuous increase in the complexity of systems demands the construction of structures with the capacity to reproduce their behavior and properties.
- Reaction to external stimuli, coordination and communication in real time challenge the limits of current state based models.

Objectives

- Define a robust structure able to adapt to real-time events, embedded with reconfigurable automata properties [Gabbay, 2008], as well as the features and formalisms of timed automata [Alur and Dill, 1991], extending such structures with the notion of actions that "take time".

Reconfigurable frames

Definition (**Multi-Actions Reactive Graph** [Tinoco et al., 2024])

A Multi-Actions Reconfigurable Graph is a tuple $M = (W, Act, E, \rightarrow, \twoheadrightarrow, \mathrel{-\!\!\times}, \bar{\cdot}, w_0, \alpha_0)$ where:

- W is a set of states.
- Act is a set of actions.
- E is a set of edges.
- $w_0 \in W$ is an initial state.
- $\alpha_0 \subseteq E$ is the set of initially active edges.
- $\rightarrow \subseteq W \times Act \times W$ is a set of ground edges.
- $\twoheadrightarrow \subseteq E \times E$ is a set of activating edges.
- $\mathrel{-\!\!\times} \subseteq E \times E$ is a set of deactivating edges.
- $\bar{\cdot} : E \longrightarrow (\rightarrow \cup \twoheadrightarrow \cup \mathrel{-\!\!\times})$ is a function that defines the internal details of edges.

Example

Reactive vending machine

- This simple example illustrates the modelization of a coffee vending machine.
- The hyper-level edges allow the machine to activate or deactivate the process depending on the selection made by the user (coffee or chocolate).

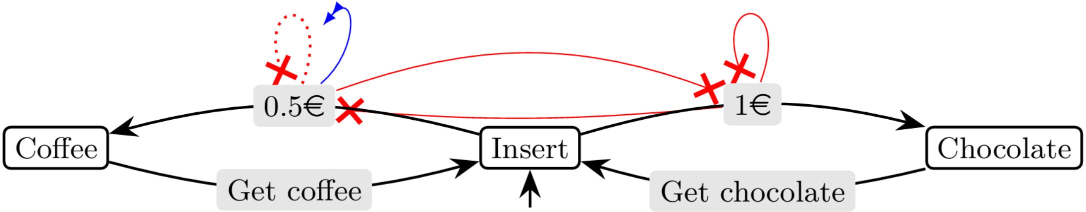
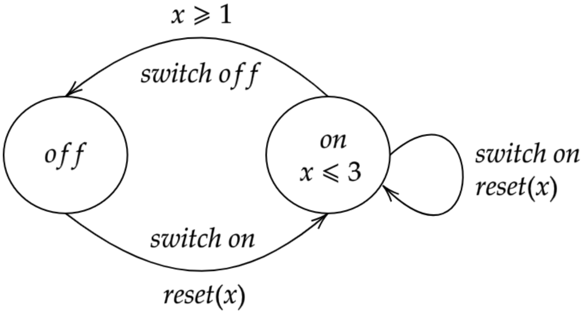


Figure: Light switch mechanism

The need of time

Reactive vending machine

- The continuous increase in the complexity of systems demands the construction of structures with the capacity to reproduce their behavior and properties.
- Real-time systems enrich the classical notion of transition systems in terms of expressiveness and complexity.



ReTA Definition

Definition

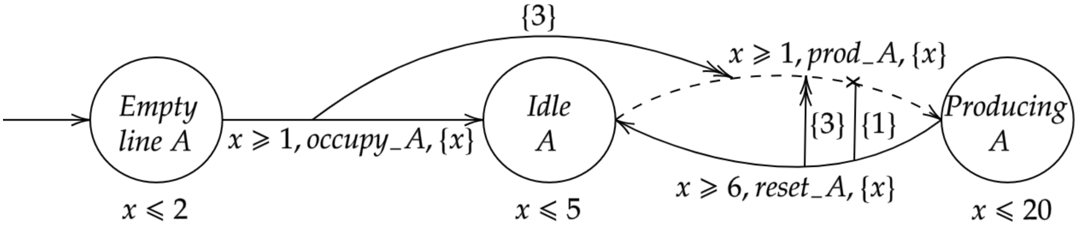
A *reconfigurable timed automata - (ReTA)* is a tuple $RT = (L, C, Act, E, \twoheadrightarrow_T, \rightarrow\!\!\times_T, Inv, R, \alpha_0, \ell_0)$ where:

- L is the *finite set of locations* and l_0 is the *initial location*.
- C is the *finite set of clocks*.
- Act is the *finite the set of actions*.
- $E \subseteq L \times CC(C) \times Act \times 2^C \times L$ is the set of *ground edges*.
- $\rightarrow\!\!\times_T \subseteq HE$ is the set of *deactivation edges* and $\twoheadrightarrow_T \subseteq HE$ is the set of *activating edges*. The set of *hyper-level edges* is given by the disjoint union $H = \twoheadrightarrow_T \uplus \rightarrow\!\!\times_T$.[1]
- $Inv : L \rightarrow CC(C)$ is the *invariant assignment function to states*
- $R : HE \rightarrow \mathbb{N}_{\geq 0}$ assigns a time delay to perform a reconfiguration.
- $\alpha_0 = (\alpha_0^E, \alpha_0^{\twoheadrightarrow}, \alpha_0^{\rightarrow\!\times})$ is *the set of initial edges*, where $\alpha_0^E \subseteq E$, $\alpha_0^{\twoheadrightarrow} \subseteq \twoheadrightarrow$ and $\alpha_0^{\rightarrow\!\times} \subseteq \rightarrow\!\!\times$ are the sets of initial active *ground edges, activation and deactivation edges*, respectively.

Example

Timed Reconfigurable Supply Chain

- This example illustrates the modelization of a supply chain single line.
- The timed hyper-level edges regulate the production process of the supply chain, controlling its procedures and protocols via timed conditions.



Configuration of a ReTA

Definition (Configurations of a ReTA)

Let a ReTA, then a configuration of a ReTA is an quadruple:

$$\Psi = (s, \eta, \Omega, \alpha)$$

where:

- $s \in S$ is the current location of the system.
- $\eta : X \to \mathbb{R}_{\geq 0}$, assigning to each clock $x \in X$ its current value $\eta(x)$.
- $\Omega \subseteq HE \times \mathbb{R}_{\geq 0}$ is the set of hyper-level edges that are in the *pool of effects* with the waiting time to be applied . We define $\Omega^{\leq d} := \{(x, t) \in \Omega : t \leq d\}$.
- $\alpha = (\alpha^E, \alpha^{\twoheadrightarrow}, \alpha^{\rightarrow\!\times})$, $\alpha^E \subseteq E$, $\alpha^{\twoheadrightarrow} \subseteq \twoheadrightarrow$ and $\alpha^{\rightarrow\!\times} \subseteq \rightarrow\!\times$ is the current set of active edges.

Transition Semantics: Delay

Definition

Let $RT = (L, C, Act, E, \twoheadrightarrow_T, \rightarrowtimes_T, Inv, R, \alpha_0, \ell_0)$ be a ReTA and $d \in \mathbb{R}_{\geq 0}$. Then $(\ell, \eta, \Omega, \alpha) \xrightarrow{d} (\ell', \eta', \Omega', \alpha')$ is a *delay transition* over RT if:

1. $\ell' = \ell$.
2. for any $x \in C$, $\eta'(x) = \eta(x) + d$
3. $\eta' \models Inv(\ell')$.
4. $\Omega' = \{(x, t) : (x, m) \in \Omega \big\backslash \Omega^{\leq d} \wedge t = m - d\}$.
5. $\alpha' = Rec(\alpha, \Omega, d)$ where $Rec(\alpha, \Omega, d) = (\alpha'^{E}, \alpha'^{\twoheadrightarrow}, \alpha'^{\rightarrowtimes})$

Transition Semantics: Discrete

Definition

Let $RT = (L, C, Act, E, \twoheadrightarrow_T, \rightarrowtail_T, Inv, R, \alpha_0, \ell_0)$ be a ReTA and $d \in \mathbb{R}_{\geq 0}$. Then, for any $a \in Act$, $(\ell, \eta, \Omega, \alpha) \xrightarrow{a} (\ell', \eta', \Omega', \alpha')$ is a *discrete transition over RT*, if:

1. There is a transition $e \in \alpha^E$, $\overline{e} : \ell \xrightarrow{g:a,D} \ell'$.
2. $\eta'(x) = \begin{cases} 0, & x \in D \\ \eta(x), & x \notin D \end{cases}$
3. $\eta' \models Inv(\ell')$.
4. $\eta \models g$.
5. $\Omega' = \Omega \cup \{(x, R(x)) \mid \pi_1(x) = e,\ x \in \alpha,\ R(x) > 0\}$.
6. $\alpha' = Rec(\alpha, \{(x, R(x)) \mid \pi_1(x) = e,\ x \in \alpha,\ R(x) = 0\}, 0)$.

Definition (**Timed Path of a ReTA**)

A path of a ReTA is an alternating sequence of configurations and timed actions:

$$\Pi = (s_0, \eta, \alpha, \Omega) \xrightarrow{(t_0, a_0)} (s_1, \eta', \alpha', \Omega') \xrightarrow{(t_1, a_1)} \ldots$$

where the pair $act_i = (t_i, a_i), i \in \mathbb{N}$ is a timed action such that:

$$\Pi = (s_0, \eta, \alpha, \Omega) \xrightarrow{t_0} (s_0, \eta + t_0, \alpha^*, \Omega^*) \xrightarrow{a_0} (s_1, \eta', \alpha', \Omega') \ldots$$

Definition (**Timed Trace of a ReTA**)

The timed trace of a ReTA (also known as timed word) is a sequence of timed actions $\lambda = (t_0, a_0)(t_1, a_1) \ldots$ from a path Π.

Definition (**Actions Trace of a ReTA**)

The actions trace of a ReTA is a sequence of discrete actions $\alpha = a_0 a_1 a_2 \ldots$ derived from a path Π, obtained by removing the timing information from its timed trace.

Definition (**Total time of a Timed Trace**)

Given a timed trace $\lambda = (t_0, a_0)(t_1, a_1)...(t_i, a_i)$ we define the total time of a Timed Trace as:

$$ExecTime(\lambda) = \sum_{i=0}^{\infty} ExecTime(\tau_i)$$

where we define the function $ExecTime : Act' \longrightarrow \mathbb{R}_{\geq 0}$ as:

$$ExecTime(\tau) = \begin{cases} 0 & \text{if } \tau \in Act \\ \tau & \text{if } \tau = d \in \mathbb{R}_{\geq 0} \end{cases}$$

Communication Protocols

Definition (**Transition Relation for composed ReTA**)

Let RT_1 and RT_2 be two ReTA and Ψ_i, Ψ_i' configurations of RT_i, $i = \{1, 2\}$.

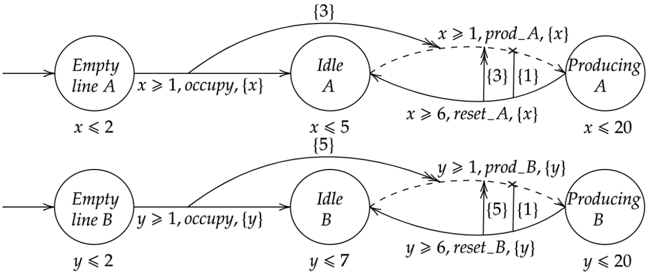
- If $\mathbf{a} \notin Act_1 \cap Act_2$, and $\Psi_1 \xrightarrow{a} \Psi_1'$ a transition of M_1, then $\langle \Psi_1, \Psi_2 \rangle \xrightarrow[RT_1 \otimes RT_2]{a} \langle \Psi_1', \Psi_2 \rangle$ is a transition of $RT_1 \otimes RT_2$.
- If $\mathbf{a} \notin Act_1 \cap Act_2$, and $\Psi_2 \xrightarrow{a} \Psi_2'$ a transition of RT_2, then $\langle \Psi_1, \Psi_2 \rangle \xrightarrow[RT_1 \otimes RT_2]{a} \langle \Psi_1', \Psi_2 \rangle$ is a transition of $RT_1 \otimes RT_2$.
- If $\mathbf{a} \in Hand$, $\Psi_1 \xrightarrow{a} \Psi_1'$ is a transition of RT_1 and $\Psi_2 \xrightarrow{a} \Psi_2'$ is a transition of RT_2, then $\langle \Psi_1, \Psi_2 \rangle \xrightarrow[RT_1 \otimes RT_2]{a} \langle \Psi_1', \Psi_2' \rangle$ is a transition of $RT_1 \otimes RT_2$.
- Let $\mathbf{d} \in \mathbb{R}_{\geq 0}$ a delay, then we have that:

$$\langle (\ell_1, \eta_1, \Omega_1, \alpha_1), (\ell_2, \eta_2, \Omega_2, \alpha_2) \rangle \xrightarrow[RT_1 \otimes RT_2]{d} \langle (\ell_1, \eta_1 + d, \Omega_1', \alpha_1'), (\ell_2, \eta_2 + d, \Omega_2', \alpha_2') \rangle.$$

Example

Composed Timed Reconfigurable Supply Chain

- This example illustrates the modelization of a supply chain single with two lines. Lines might be operated by one worker (*handshaking*) or independently by different workers (*parallel*).



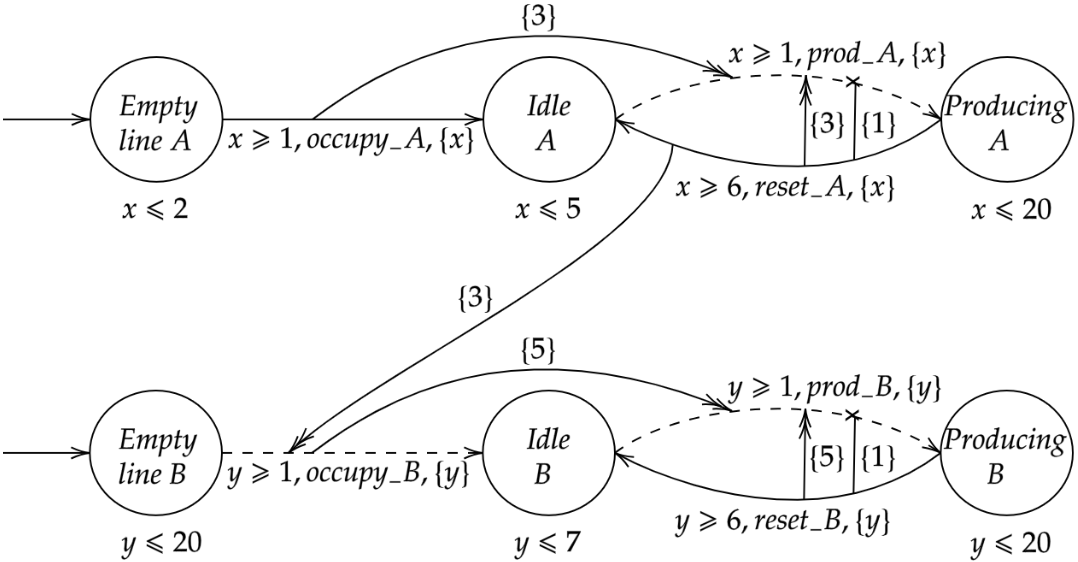
Communication Protocols: Timed Intrusive

Definition (**Transition Relation for composed intrusive ReTA**)

Let RT_1 and RT_2 be two ReTA and Ψ_i, Ψ'_i configurations of RT_i, $i = \{1, 2\}$ and the sets of intrusive transitions $\Gamma_1 \subseteq E_1 \times (E_2 \cup HE_2)$ and $\Gamma_2 \subseteq E_2 \times (E_1 \cup HE_1)$.

- If **a** $\notin Act_1 \cap Act_2$, and $\Psi_1 \xrightarrow{a} \Psi'_1$ a transition of RT_1, then $\langle \Psi_1, \Psi_2 \rangle \xrightarrow[RT_1 \otimes RT_2]{a} \langle \Psi'_1, \Psi'_2 \rangle$ is a transition of $RT_1 \otimes RT_2$ such that $\alpha'_1 = \alpha_1 \cup Rec(\alpha_1, \Omega_1, 0)$ and $\alpha'_2 = \alpha_2 \cup Rec(\alpha_2, \Omega_1, 0)$.
- If **a** $\notin Act_1 \cap Act_2$, and $\Psi_2 \xrightarrow{a} \Psi'_2$ a transition of RT_2, then $\langle \Psi_1, \Psi_2 \rangle \xrightarrow[RT_1 \otimes RT_2]{a} \langle \Psi'_1, \Psi'_2 \rangle$ is a transition of $RT_1 \otimes RT_2$ such that $\alpha'_1 = \alpha_1 \cup Rec(\alpha_1, \Omega_2, 0)$ and $\alpha'_2 = \alpha_2 \cup Rec(\alpha_2, \Omega_2, 0)$.
- If **a** $\in Hand$, $\Psi_1 \xrightarrow{a} \Psi'_1$ is a transition of RT_1 and $\Psi_2 \xrightarrow{a} \Psi'_2$ is a transition of RT_2, then $\langle \Psi_1, \Psi_2 \rangle \xrightarrow[RT_1 \otimes RT_2]{a} \langle \Psi'_1, \Psi'_2 \rangle$ is a transition of $RT_1 \otimes RT_2$, such that $\alpha'_1 = \alpha_1 \cup Rec(\alpha_1, \Omega_2, 0) \cup Rec(\alpha_1, \Omega_1, 0)$ and $\alpha'_2 = \alpha_2 \cup Rec(\alpha_2, \Omega_2, 0) \cup Rec(\alpha_2, \Omega_1, 0)$.
- Let **d** $\in \mathbb{R}_{\geq 0}$ a delay, then we have that: $\langle (\ell_1, \eta_1, \Omega_1, \alpha_1), (\ell_2, \eta_2, \Omega_2, \alpha_2) \rangle \xrightarrow[RT_1 \otimes RT_2]{d} \langle (\ell_1, \eta_1 + d, \Omega'_1, \alpha'_1), (\ell_2, \eta_2 + d, \Omega'_2, \alpha'_2) \rangle$.

Example


{3}
$x \geqslant 1, prod\_A, \{x\}$
Empty line A
$x \geqslant 1, occupy\_A, \{x\}$
Idle A
{3}
{1}
Producing A
$x \geqslant 6, reset\_A, \{x\}$
$x \leqslant 2$
$x \leqslant 5$
$x \leqslant 20$
{3}
{5}
$y \geqslant 1, prod\_B, \{y\}$
Empty line B
$y \geqslant 1, occupy\_B, \{y\}$
Idle B
{5}
{1}
Producing B
$y \geqslant 6, reset\_B, \{y\}$
$y \leqslant 20$
$y \leqslant 7$
$y \leqslant 20$

Existing tools

Motivation

- Lack of specialized software for reconfigurable systems.
- MARGe [Tinoco et al., 2024] software: Modeling and verification.
- Uppaal [Behrmann et al., 2004] software: Suitable tool for timed systems, CTL verifier, graphical interface...
- Need to homogenize the functionalities of both reconfigurable and timed automata.

HASKELL encoding

Program functionalities [Iglesias, 2025]

- Four steps definition:
  1. Composed system (Y/N)?
  2. Automata element definition: Locations, guards, invariants, edges.
  3. Shared actions (Y/N)?
  4. Hyper-edges definition.
- Simulation of paths:
  1. Delay transitions.
  2. Discrete transitions. Includes shared actions.
- Trace output and total execution time.
- Detection of misbehavior:
  1. Deadlock violation.
  2. Non-verification of guards.
  3. Non-active transitions.

Timed runs in HASKELL

```
--- Mantra M1 ---
  Loc: EmptyA, Clocks: [("x",0)], Omega: [], ActiveEdges: fromList ["Edge1","Edge3"], ActiveHyper: fromList ["HE1"]
-----------------------------
```

```
--- Mantra M1 ---
  Loc: IdleA, Clocks: [("x",1)], Omega: [("HE1",2)], ActiveEdges: fromList ["Edge1","Edge3"], ActiveHyper: fromList ["HE1"]
-----------------------------
```

```
--- Mantra M1 ---
  Loc: IdleA, Clocks: [("x",5)], Omega: [], ActiveEdges: fromList ["Edge1","Edge2","Edge3"], ActiveHyper: fromList []
-----------------------------
```

(a) Initial configuration | After first discrete transition and 1-unit delay | After hyper-edge effects

```
--- Composed System State ---
Shared Actions: []
Total Execution Time: 14 time units
Trace:
  Delay Action: 1 time units
  Discrete Action: 'ocA' by Mantra 'M1'
  Delay Action: 1 time units
  [FAIL] Discrete transition failed for action: prodA by mantra: M1. Shared status: False
  Delay Action: 4 time units
  Discrete Action: 'prodA' by Mantra 'M1'
  Delay Action: 5 time units
  [FAIL] Discrete transition failed for action: resA by mantra: M1. Shared status: False
  [FAIL] TIMELOCK PREVENTED: Delay of 21 violates an invariant in at least one mantra.
  Delay Action: 3 time units
  Discrete Action: 'resA' by Mantra 'M1'
```

(b) Complete trace of actions

Shared actions run

```
--- Current System State ---

--- Composed System State ---
Shared Actions: ["oc"]
Total Execution Time: 2 time units
Trace:
  Delay Action: 2 time units
  Shared Discrete Action: 'oc'


--- Mantra M2 ---
  Loc: IdleB, Clocks: [("y",0)], Omega: [("HE2",5)], ActiveEdges: fromList ["Edge4","Edge6"], ActiveHyper: fromList ["HE2"]
--- Mantra M1 ---
  Loc: IdleA, Clocks: [("x",0)], Omega: [("HE1",3)], ActiveEdges: fromList ["Edge1","Edge3"], ActiveHyper: fromList ["HE1"]
-----------------------------
```

```
--- Composed System State ---
Shared Actions: ["oc"]
Total Execution Time: 8 time units
Trace:
  Delay Action: 2 time units
  Shared Discrete Action: 'oc'
  Delay Action: 3 time units
  Discrete Action: 'prodA' by Mantra 'M1'
  [FAIL] Discrete transition failed for action: prodB by mantra: M2. Shared status: False
  Delay Action: 3 time units
  Discrete Action: 'prodB' by Mantra 'M2'


--- Mantra M2 ---
  Loc: ProdB, Clocks: [("y",0)], Omega: [], ActiveEdges: fromList ["Edge4","Edge5","Edge6"], ActiveHyper: fromList []
--- Mantra M1 ---
  Loc: ProdA, Clocks: [("x",3)], Omega: [], ActiveEdges: fromList ["Edge1","Edge2","Edge3"], ActiveHyper: fromList []
-----------------------------
```

Intrusive actions run

```
--- Current System State ---

--- Composed System State ---
Shared Actions: []

--- Mantra M2 ---
  Loc: EmptyB, Clocks: [("y",11)], Omega: [], ActiveEdges: fromList ["Edge6"], ActiveHyper: fromList ["HE2","HE3"]
--- Mantra M1 ---
  Loc: IdleA, Clocks: [("x",0)], Omega: [("HE2",3)], ActiveEdges: fromList ["Edge1","Edge2","Edge3"], ActiveHyper: fromList ["HE2"]
-----------------------------
```

```
--- Current System State ---

--- Composed System State ---
Shared Actions: []

--- Mantra M2 ---
  Loc: EmptyB, Clocks: [("y",14)], Omega: [], ActiveEdges: fromList ["Edge4","Edge6"], ActiveHyper: fromList ["HE2","HE3"]
--- Mantra M1 ---
  Loc: IdleA, Clocks: [("x",3)], Omega: [], ActiveEdges: fromList ["Edge1","Edge2","Edge3"], ActiveHyper: fromList []
-----------------------------
```

Conclusions and future work

Future objectives

- Graphical interface.
- Upload/Download models.
- Enrich the encoding with new theoretical components, i.e, model checking, suitable logics...
- Extend the ReTA models into other suitable software.
- Reformulate ReTA into bisimilar structures.

REFERENCES

Alur, R. and Dill, D. L. (1991).
The theory of timed automata.
In de Bakker, J. W. et al., editors, *Real-Time: Theory in Practice, REX Workshop*, volume 600 of *Lecture Notes in Computer Science*, pages 45–73. Springer.

Behrmann, G., David, A., and Larsen, K. G. (2004).
A tutorial on uppaal.
In *Formal Methods for the Design of Real-Time Systems*, pages 200–236. Springer.

Gabbay, D. M. (2008).
Introducing reactive kripke semantics and arc accessibility.
In Avron, A. et al., editors, *Pillars of Computer Science*, pages 292–341. Springer.

Iglesias, A. C. C. (2025).
Haskell reta encoding.
`https://gitlab.com/antonio.iglesias13/haskell_reta_encoding`.
GitLab repository.

Tinoco, D., Madeira, A., Martins, M. A., and Proença, J. (2024).
Reactive graphs in action.
In Marmsoler, D. and Sun, M., editors, *FACS 2024*, volume 15189 of *Lecture Notes in Computer Science*, pages 97–105. Springer.