Université Paris Sorbonne Nord
Institut Galilée

INFO-2A.
Functional Programming
Mini-Project

In this assignment you will apply the functional programming techniques we have learned during the semester. It is crucial that you demonstrate a solid understanding of the functional paradigm. To achieve this, define data types and create concise and well-defined functions (with explicit type annotations) that operate on them. You are free, and encouraged, to use the modules defined in the standard library of Haskell in your code.

Deadline: October 04, 16:00. Submit a unique Zip file on Moodle. You can work in pairs. Do not forget to add the name of the two students in a Readme file (and submit only one file per team). Collaboration between teams is strictly forbidden. Any similarity between codes will be considered plagiarism and will result in severe penalties

1 Introduction

Reaction Systems (RSs) [BEMR11] are a formal framework inspired by the functioning of living cells and by biochemistry. RSs have proven to be a versatile computational model with a wide range of applications, including the modeling of biological processes [AIP14, CMM$^+$12, Azi17, BGLM16], molecular chemistry [OY16], and computer science systems [BEMR11, EMR10, EMR11]. A RS comprises a finite set of entities and a finite set of reactions acting on the entities. Each reaction is described by a triple (R, I, P), where R denotes *reactants* (entities that must be present for the reaction to occur), I denotes the *inhibitors* (entities that must be absent to enable the reaction), and P denotes the *products* (entities generated if the reaction takes place).
The behavior of a RS is typically specified as a rewrite system that models *processes interacting* with an external *context*. The external context specifies the entities provided by the environment at each step of the process. The current system state is determined by the union of the entities coming from the environment with those produced in the previous state. A state transition is then determined by applying all and only the enabled reactions in the current state.
In this project we will define the needed data types and functions to simulate a reaction system interacting with a possibly non-deterministic environment, and verify some properties about it.

2 Reaction Systems

We use the term *entities* to denote generic molecular substances (e.g., atoms, ions, molecules) that may be present in the states of a biochemical system. The main mechanisms that regulate the functioning of a living cell are *facilitation* and *inhibition*. These mechanisms are based on the presence and absence of entities and are reflected in the basic definitions of RSs.

Definition 1 (Reaction Systems)**.** *Let S be a (finite) set of entities. A reaction in S is a triple $a = (R, I, P)$, where $R, I, P \subseteq S$ are finite sets and $R \cap I = \emptyset$. We use $rac(S)$ to denote the set of all reactions on the set of entities S. A Reaction System (RS) is a pair $\mathcal{A} = (S, A)$ s.t. S is a finite set of entities, and $A \subseteq rac(S)$ is a finite set of reactions.*

The sets R, I, P are the sets of *reactants*, *inhibitors*, and *products*, respectively. Due to biological considerations, the sets R and P are usually nonempty: there is no creation from nothing ($R \neq \emptyset$), and if a reaction takes place then something is produced ($P \neq \emptyset$). A reaction can take place whenever all of its reactants are present in a given state while none of its inhibitors is present (predicate $en_a(\cdot)$ below). If this happens, the reaction is enabled and creates its products (function $res_A(\cdot)$ below). More formally,

Definition 2 (Reaction Result)**.** *Given a (finite) set of entities S, and a subset $W \subseteq S$, we define the following:*

1. *Let $a = (R, I, P) \in rac(S)$ be a reaction in S. The result of reaction a on W, denoted by $res_a(W)$, is defined by:*

$$res_a(W) \triangleq \begin{cases} P & \text{if } en_a(W) \\ \emptyset & \text{otherwise} \end{cases}$$

   where the enabling *predicate is defined by*

$$en_a(W) \triangleq (R \subseteq W) \wedge (I \cap W = \emptyset)$$

2. *Let $A \subseteq rac(S)$ be a finite set of reactions. The result of A on W, denoted by $res_A(W)$, is defined by: $res_A(W) \triangleq \bigcup_{a \in A} res_a(W)$.*

Reaction Systems are based on three assumptions:

1. **No permanency**, i.e., any entity vanishes unless it is sustained by a reaction. Therefore, if an entity in a system state is not supported by at least one reaction which produces it, then it will disappear in the following state.

2. **No counting**, the basic model of RSs is very abstract and qualitative, i.e. the quantity of entities that are present in a cell is not considered.

3. **Threshold nature of resources**, it is assumed that either an entity is available for all reactions, or it is not available at all. In other terms, two or more reactions that are enabled by a set of entities W always generate their products, even if they share some entities, because there is always a sufficient amount of reactants in W to activate all the enabled reactions.

The previous assumptions imply that nonempty intersection of sets of reactants of enabled reactions never generates conflicts. For instance, given the set of reactions $A = \{a_1, a_2\}$, where $a_1 = (\{\mathsf{a}, \mathsf{b}\}, \{\mathsf{c}\}, \{\mathsf{b}\})$ and $a_2 = (\{\mathsf{a}\}, \{\mathsf{d}\}, \{\mathsf{c}\})\}$, both a_1 and a_2 are enabled on the set of entities $D = \{\mathsf{a}, \mathsf{b}\}$. Moreover, both take place simultaneously, producing $\{\mathsf{b}\}$ and $\{\mathsf{c}\}$.

Since living cells are seen as open systems that interact with the external environment, the behavior of a RS is formalized through the notion of *interactive process*, that is, a process that react to external stimuli.

Definition 3 (Interactive Process)**.** *Let $\mathcal{A} = (S, A)$ be a RS and let $n \geq 0$. An n-step* interactive process *in $\mathcal{A}$ is a pair $\pi = (\gamma, \delta)$ s.t. $\gamma = \{C_i\}_{i\in[0,n]}$ is the* context sequence *and $\delta = \{D_i\}_{i\in[0,n]}$ is the* result sequence*, where $C_i, D_i \subseteq S$ for any $i \in [0, n]$, $D_0 = \emptyset$, and $D_{i+1} = res_A(D_i \cup C_i)$ for any $i \in [0, n-1]$. We call $\tau = W_0, \ldots, W_n$ with $W_i \triangleq C_i \cup D_i$, for any $i \in [0, n]$ the* state sequence.

The context sequence γ represents the environment interacting with the RS. It specifies the *input* that comes from the environment. The result sequence δ is entirely determined by γ and the reactions in A. As expected, distinct context sequences may lead to distinct outcomes for the same reaction set A, but a context sequence γ determines a *unique* result sequence δ.

Below an example of a part of a RS modeling signaling networks in breast cancer taken from [dHBH$^+$14, BBFO24].

Example 1. *Let $\mathcal{A} = (S, A)$ be the RS where*

$$\begin{array}{lcl} S & = & \{\mathsf{egf}, \mathsf{e}, \mathsf{p}, \mathsf{erbb1}, \mathsf{erk12}, \mathsf{p70s6k}\} \\ A & = & \{a_1, a_2, a_3\} \\ a_1 & = & (\{\mathsf{egf}\}, \{\mathsf{e}, \mathsf{p}\}, \{\mathsf{erbb1}\}) \\ a_2 & = & (\{\mathsf{egf}\}, \emptyset, \{\mathsf{erk12}\}) \\ a_3 & = & (\{\mathsf{erk12}\}, \emptyset, \{\mathsf{p70s6k}\}) \end{array}$$

Consider the context sequence $\gamma = C_0, \cdots, C_3$, with $C_0 = \{\mathsf{egf}, \mathsf{e}\}$, $C_1 = \emptyset$, $C_2 = \{\mathsf{egf}\}$, and $C_3 = \emptyset$. Then, $\pi = (\gamma, \delta)$ in an interactive process where $\delta = D_0, \cdots, D_3$, with $D_0 = \emptyset$, $D_1 = \{\mathsf{erk12}\}$, $D_2 = \{\mathsf{p70s6k}\}$, and $D_3 = \{\mathsf{erbb1}, \mathsf{erk12}\}$. The resulting state sequence is $\tau = W_0, \cdots, W_3$, where $W_0 = \{\mathsf{egf}, \mathsf{e}\}$, $W_1 = \{\mathsf{erk12}\}$, $W_2 = \{\mathsf{egf}, \mathsf{p70s6k}\}$, and $W_3 = \{\mathsf{erbb1}, \mathsf{erk12}\}$.

3 Context and Interaction

As explained above, RSs interact with a context providing as input some entities $D \subseteq S$. We call D the set of entities controlled by the environment. Below we present the syntax of a simple language to define processes that can interact with a given RS $\mathcal{A} = (S, A)$ (by providing input contexts to it).

Definition 4 (Processes)**.** *The language of processes is defined as:*

$$K, M \quad ::= \quad \mathbf{0} \quad | \quad X \quad | \quad \textstyle\sum_{i\in I} C_i.K_i \quad | \quad K \parallel M \quad | \quad \mathbf{rec}\ X : K$$

where X is a process variable, I is a finite nonempty set of indices, and $C \subseteq S$ is a possibly empty set of entities.

Below we give an intuitive description of the constructors of this language.
A process $C.K$, where $C \subseteq S$ and K is a process, represents the situation where the current stimulus to the RS is the set of entities C. Then, in the next interaction, the stimulus will be determined by K. The process $\mathbf{0}$ represents the end of a context sequence. The recursive definition $\mathbf{rec}\ X : K$ is unfolded and the process $K[\mathbf{rec}\ X : K/X]$ is executed. For instance,

the process **rec** $X : C.X$ unfolds into $C.(\textbf{rec } X : C.K)$, thus producing the context sequence $C.C.C.C.\cdots$. We assume that all occurrences of X in **rec** $X : K$ appear in the scope of a prefix (e.g., **rec** $X : C.X$ is guarded while **rec** $X : X$ is not). Moreover, all the occurrences of X in a process K must be bound by the binder "**rec** X :" (i.e., there are no free variables in a well-formed process)[1].
The process $\sum_{i \in I} C_i.K_i$ chooses, non-deterministically, one of the sets C_i and precludes the others from execution. To simplify the notation: when the set of indices I is a singleton, we omit the "$\sum_{i \in I}$" part and we write $C.K$; if $|I| = 2$, we write $C_1.K_1 + C_2.K_2$.
The parallel composition of two processes is represented by $K \parallel M$, where the outputs of K and M are merged together.

4 Description of the Project

Implement the needed data types and functions to:

1. Observe the output of a RS under a given input $C_1.C_2.\cdots$.
2. Observe the behavior of a RS when interacting with a process K.
3. Check whether a given entity is produced in a RS when interacting with a process K. For instance, the entity `d` is produced in the following RS and process:

```
( a ; b ; c )
( c ; a ; d )

proc : rec X . (a.X + b.X)
```

4. Check whether the RS stabilizes, i.e., there is a cycle (loop) in the system where a given state is visited infinitely often. For instance, in the previous example, we have the cycle below, where: `input` is the output in the previous time-unit (`empty` for the first time-unit); `proc` is the process being executed; `context` is the set of entities produced by the process in the current time-unit; and `output` is the set of entities produced in the current interaction.

```
input: empty , proc: rec X. (a.X + b.X), context: a, output: c
input: c     , proc: rec X. (a.x + b.X), context: b, output: d
input: d     , proc: rec X. (a.x + b.X), context: a, output: c
input: c     , proc: rec X. (a.x + b.X), context: b, output: d
```

For getting a better score, consider the following requirements. Define a simple language for expressing properties of the system including:

1. Atomic propositions: e (the entity e is present) and $\neg e$ (the entity e is not present in the output)
2. Propositional symbols: $\wedge$ and $\vee$. For instance, $e_1 \wedge \neg e_2$ means that e_1 is present and e_2 is absent.

[1] If you do not find these restrictions reasonable, please ask!

3. Eventually: $\Diamond\phi$, where ϕ is a formula built from atomic propositions and propositional symbols, means that ϕ eventually holds. Note that checking whether the entity e is produced can be expressed with the formula $\Diamond e$ (there is a path in the execution where e is eventually present.

4. Always: $\Box\phi$, where ϕ is a formula built from atomic propositions and propositional symbols, means that ϕ always holds. For instance, $\Box e$ means that the system stabilizes in a state where e is present.

5. Until, $\phi \mathbf{U} \phi'$ where ϕ and ϕ' are as above, meaning that ϕ' eventually holds and, before that ϕ remains true. For instance, the formula $e\mathbf{U}e'$ means that there is a path of the form:

$$s_1.s_2.\cdots.s_n$$

where e' is present in s_n and e is present in s_i for $1 \leq i < n$.

6. Can you handle ...

The RS and the process to be executed must be read from an text file.
If you want to test your code with real RSs, send me an email!

References

[AIP14] Sepinoud Azimi, Bogdan Iancu, and Ion Petre. Reaction system models for the heat shock response. *Fundam. Informaticae*, 131(3-4):299–312, 2014.

[Azi17] Sepinoud Azimi. Steady states of constrained reaction systems. *Theor. Comput. Sci.*, 701(C):20–26, 2017.

[BBFO24] Demis Ballis, Linda Brodo, Moreno Falaschi, and Carlos Olarte. Process calculi and rewriting techniques for analyzing reaction systems. In Roberta Gori, Paolo Milazzo, and Mirco Tribastone, editors, *Computational Methods in Systems Biology*, pages 1–18, Cham, 2024. Springer Nature Switzerland.

[BEMR11] R. Brijder, A. Ehrenfeucht, M. Main, and G. Rozenberg. A tour of reaction systems. *Int. J. Found. Comput. Sci.*, 22(07):1499–1517, 2011.

[BGLM16] Roberto Barbuti, Roberta Gori, Francesca Levi, and Paolo Milazzo. Investigating dynamic causalities in reaction systems. *Theor. Comput. Sci.*, 623:114–145, 2016.

[CMM+12] Luca Corolli, Carlo Maj, Fabrizio Marinia, Daniela Besozzi, and Giancarlo Mauri. An excursion in reaction systems: From computer science to biology. *Theor. Comput. Sci.*, 454:95–108, 2012.

[dHBH+14] Silvia Von der Heyde, Christian Bender, Frauke Henjes, Johanna Sonntag, Ulrike Korf, and Tim Beißbarth. Boolean ErbB network reconstructions and perturbation simulations reveal individual drug response in different breast cancer cell lines. *BMC Systems Biology*, 8(1):75, 2014.

[EMR10] Andrzej Ehrenfeucht, Michael G. Main, and Grzegorz Rozenberg. Combinatorics of life and death for reaction systems. *Int. J. Found. Comput. Sci.*, 21(3):345–356, 2010.

[EMR11] Andrzej Ehrenfeucht, Michael G. Main, and Grzegorz Rozenberg. Functions defined by reaction systems. *Int. J. Found. Comput. Sci.*, 22(1):167–178, 2011.

[OY16] Fumiya Okubo and Takashi Yokomori. The computational capability of chemical reaction automata. *Natural Computing*, 15(2):215–224, 2016.