

Theoretical Computer Science

A process algebraic approach to reaction systems.

--Manuscript Draft--

Manuscript Number:	TCS-D-20-00267R1
Article Type:	SI: RS
Section/Category:	C - Theory of natural computing
Keywords:	Process algebras; reaction systems; Assertion language; Hennessy-Milner logic.
Corresponding Author:	Moreno Falaschi University of Siena Siena, ITALY
First Author:	Linda Brodo, Researcher
Order of Authors:	Linda Brodo, Researcher
	Roberto Bruni, Professor
	Moreno Falaschi, Professor
Manuscript Region of Origin:	ITALY
Abstract:	<p>In the area of Natural Computing, Reaction Systems (RSs) are a qualitative abstraction inspired by the functioning of living cells, suitable to model the main mechanisms of biochemical reactions. RSs interact with a context, and pose challenges for modularity, compositionality, extendibility and behavioural equivalence. In this paper we define a modular encoding of RSs as processes in the chained Core Network Algebra (cCNA), which is a new variant of the link-calculus. The encoding represents the behaviour of each entity separately and preserves faithfully their features, and we prove its correctness and completeness. Our encoding provides a Labelled Transition System (LTS) semantics for RSs. Based on the LTS semantics, we adapt the classical notion of bisimulation to define a novel equivalence, called bio-similarity, for studying properties of RSs. In particular, we define a new assertion language based on regular expressions, which allows us to specify the properties of interest, and use it to extend Hennessy-Milner logic to our setting. We prove that our bio-similarity relation and the logical equivalence, that are defined parametrically on some assertion of interest, coincide. Finally, we claim that our encoding contributes to increase the expressiveness of RSs, by exploiting the interaction among different RSs.</p>

Dear Editor and Referees,

we thank you very much for your detailed comments and remarks, which greatly helped us to correct and improve our presentation.

We updated the manuscript following the advices of the referees. Please find below a detailed description (in blue) of the way in which the items pointed out by the referees were addressed in the revised version.

Best regards,

Linda Brodo, Roberto Bruni, Moreno Falaschi

Manuscript Number: TCS-D-20-00267

Manuscript Title: A process algebraic approach to reaction systems.

Author(s): Linda Brodo, Researcher; Roberto Bruni, Professor; Moreno Falaschi, Professor

Submitted to: Theoretical Computer Science

Reviewer #1: In this paper the authors consider a variant of the link-calculus, and show that one can represent reaction systems. For this embedding they show its correctness and completeness. Several examples are use to show how the embedding works. Also, they define an assertion language based on regular expressions, and use it to extend the Hennessy-Milner logic to their framework. They prove that their notion of bisimilarity and the logical equivalence coincide. They also claim to increase the expressiveness of reaction systems, by exploiting the interaction among different reaction systems.

The paper seems technical sound, but the long chains are not easy to read and understand. For example I couldn't follow how the chain given in Figure 2 in obtain from the given interacting processes. Maybe a more detailed connection should be made between the depicted chains, those on labels and those used in the given processes.

> We have improved the description of the example and Figure 2 to make the correspondence between processes and transition labels more evident.

Also, some processes used in examples seem not to follow the definitions given on pages 9 and 10 for embedding rules and objects. Why?

> For the sake of readability, when possible, due to the simple examples in hand, we have chosen to give a simplified version of the process definition. Thus, it is not the case that some processes do not follow the definitions on pages 9 and 10, the fact is that we have discarded those parts of the code that would not be executed in that specific example.

Remarks:

p.2, l.52 - replace "looks a suitable option" by "is a suitable option"

> Done.

p.3, l.39 - you mention here that interactive reaction systems do not exist, but this is no longer the case. See Networks of Reaction

Systems <https://doi.org/10.1142/S0129054120400043>

>Right, we added a sentence citing the recent Reaction System extension you suggested.

p.8, Fig.1 - you mention that rule (Rel) is omitted. What does this rule do and where is it used in the paper? This rule isn't explained on the previous page with the other rules of Fig.

> Alike CCS, CNA includes a relabelling operator that is helpful to reuse the same process in different contexts. Relabelling is not needed to encode Reaction Systems and thus we have omitted it from cCNA. This is now explained in the text. Now all rules are reported in the figure.

p.9, l.9 - you mention "four different names" but in what follows you give and explain five pairs of names. Why?

> It was a typo, now it is "five different pairs names".

p.10, l.21 - replace "Channels p_i " by "Channels p_j "

> Done.

p.10, l.33, def. of Cxt^n - replace s_j and $\overline{e_j}$ by s_i and $\overline{e_i}$ in order to have uniform notations corresponding to the notations on top of page 9

> Actually, in the text they were s_i and $\overline{e_i}$; it was a font overlapping problem. Now it is solved.

p.11, l.29 - shouldn't be Cxt^1 instead of Cxt ? if this is the check please check all appearances of Cxt without superscript from all over the paper (e.g., example 15)

> Ok, fixed.

p.12, Fig.2. - maybe add some brackets to illustrate each part of the chain to which process corresponds. While the parts taken from the processes I and Cxt^1 are easily to spot, the same thing can't be said about Pr_1 and Pr_2 . E.G., Pr_1 should start with $r_1 \backslash s1_i$ but this is not part of Fig.2. Why?

> We added the brackets to show which process is involved in each part of the link chain.

p.12, l.28, def. of $P_{\{s1\}}$ - why are the pairs $\text{box}\backslash\text{box}\ \tilde{s1_i}\ \text{tile}\{s1_o\}$ missing? according to top of page 9 shouldn't it also be present as r_2 can produce s_1 ?

> Virtual links at the left/right extremities of chain prefixes can be always omitted. This is now explained in a remark that we added in the Section about cCNA, before the operational semantics.

p.12, l.32, def of Cxt - shouldn't it contain entities without decorations according to its definition given on page 10?

> Right ! We corrected definition of Cxt on page 10.

p.14, l.50-51 - all four rules do not describe the evolution depicted in Fig.1, namely the final states are swapped (w and q I mean). E.g., the first rule should reach w and not q as it does now

> Done.

p.15, l.22-23 - just like at the previous remark, processes P_1 and P_2 have last states swapped

> Done.

p.15, l.56 - why is the definition of Cxt different from that of Cxt^n given on page 10, namely the decorations are different?

> Fixed.

p.16, l.10 - replace "context: We model" by "context: we model" or "context. We model"

> Done.

p.16, l.28 - replace "the entity w " by "the entity q " (according to previous remark on swapped states)

> Done.

p.16, l.32 - replace the symbol \tilde{w} between p_2 and p_3 by \tilde{q}

> Done.

p.16, l.41 - something seems to be missing here as the start of the paragraph makes no sense now

> Ok, removed.

p.17, l.31 - replace $\overline{P}_{T, \tilde{T}}$ by \overline{P}_T

> Done.

p.17, l.36 - according to the definitions on page 9, should be $\boxtimes (\tilde{c}_i \tilde{c}_o) \boxtimes^h$ instead of $(\boxtimes \tilde{c}_i) \tilde{c}_o^h$?

> Right, done.

p.17, l.40 - before $\cdot p$ shouldn't be \boxtimes according to the definitions on page 9?

> The link chains with or without virtual links at their extremes are equivalent, as we have explained in the paper.

p.17, l.49-50 - remove the dots after the power h in the second processes of both lines

> Done.

p.18, l.34 - replace "when the expose" by "when they expose"

> Done.

p.12, l.12 - replace "contest" by "context"

> Done.

p.23, l.11-14 - why Cxt_1 and Cxt_2 have a different form than the definition of Cxt^n on page 10?

> We fixed the notation on pag. 10. Now, we have changed the names Cxt_1 and Cxt_2 with Cxt and Cxt' and their definitions are coherent with the one of Cxt_n on page 10.

p.23, l.18 - according to the definition on page 9, the pair \boxtimes should appear before the dot (twice); also the dot is missing in the definition of $\overline{P}(G)$; same remark for almost all processes on page 25

> The link chains with or without virtual links at their extremes are equivalent, as we have explained in the paper.

We have added the missing dot.

p.23, l.26-27 - the definitions of Sys_1 and Sys_2 seem swapped based on the explanations given before the relations. E.g., Sys_1 includes C, but in the formal part it contains the process \overline{P}_C . This swapping seems to propagate till the end of Section 6 so please check carefully that part also

> Right, fixed.

p.25 - try not to use colours as in the printed black/white version those are not visible

> Yes, you are right, but at least in the electronic version they are visible and can help the reader.

p.26, ref. [7] - remove the hyperlink from paper title, and also the URL as this is the only reference that has both a DOI and an URL

> Done

p.27, ref. [15] - replace "A. Bogdan, C. Gabriel" by "B. Aman, G. Ciobanu"; this seems to be the only place with such an error, but please check all other references as well

> Done.

p.30, l.17 - who are h and w in " $h \in [1, \dots, w]$ "? Also who is cxt_h as only Cxt^n was defined previously?

> Right, fixed. It was a typo: $\$cxt_h\$$ simply was the channel name $\$cxt\$$.

p.30, l.28, def. $P_{\{aj\}}$ - replace $\ldots \tilde{s}_o$ by $\tilde{s}_o \ldots$

> Done.

p.30, l.33, def. $P_{\{aj\}}$ replace space before s_o ; same issue with extra spaces appears in the rest of the formulae on page 30

> Done.

Reviewer #2:

Title: A process algebraic approach to reaction systems

In the first part of the paper the authors define a framework for encoding reaction systems as processes in the open multi-party process algebra cCNA (chained Core Network Algebra) and prove the correctness and the completeness of this encoding. In the second part the authors define a new bisimulation relation, called bio-simulation, alongside an assertion language extending the Hennessy-Milner logic. This paper is an extended version of the conference paper [16] and the authors listed the additional results and included proofs of the results.

In general this algebraic approach of encoding Reaction Systems is interesting and seems sound.

While I am generally in favour of the work, I believe this paper should be improved in a few ways before publication. My concerns regard the conciseness of the abstract, the introduction and the conclusions, as well as the clarity of the actual contributions. In the body of the paper more details should be added in places mentioned below such that the heavy notations are easier to grasp.

> We have rewritten abstract, introduction and conclusion according to your suggestions.

In the following I list questions, comments, suggestions, and typos.

Page1

The abstract of the paper could be made more concise. Second sentence is not needed and it's somehow contradictory with the statement of the third sentence "it is a new computation model". In the 3rd sentence, the justification of the challenges is not convincing, maybe just a simple rephrasing would help.

> Ok, rephrased.

Line 28 - say that you are using chained Core Network Algebra (cCNA) which is a variant of the link-calculus. When you say "we consider", it's not clear if you define it or just use it. Why not say simpler (instead of sentence starting at line 28): We define a (modular?) encoding/embedding of Reaction Systems as processes in the chained Core Network Algebra (cCNA), which is a variant of the link-calculus, by representing ..., and we prove its correctness and completeness.

> The text is modified accordingly.

Line 33 - "Computer science applications" sounds quite vague.

> Ok, deleted.

Line 35 - framework, embedding, encoding, approach, translation -- it is not clear all through the paper if there is a distinction between these 5 words, better to settle on 1 or 2

> Ok, we have chosen to use the word "encoding".

Line 34 - I would say it as a statement: "Our *framework* provides a Labelled Transition System (LTS) semantics for reaction systems. Based on the LTS semantics we adapt the classical notion of process bisimulation to define a bio-simulation relation for studying properties of reaction systems.

> Ok, rephrased.

Line 40 - Remove "Interestingly" and instead make the statement: We prove that our bio-simulation relation and the logical equivalence coincide.

> Ok, rephrased.

Line 42 - A stronger statement would be: "We claim/show/prove(?) that our methodology contributes to ..."

> Done.

Line 50 - Is Natural Computing still an emerging area of research?

> Ok, rephrased.

Page 2

In some place it's Reaction Systems with capital letter, in others is lower caps. Please make this consistent throughout the paper.

Line 16 - For cCNA please remove the footnote and put the full description in the text. Is this something defined in this paper or not, in which case add biblio reference(s).

> Ok, rephrased.

Line 19 - "this mechanism" refers to cCNA or to link-calculus?

> Ok, rephrased.

Line 21 - How is [12] "later" as [11] is from 2018 and [12] from 2014?

> This is due to the fact that the citation [11] ,now [], refers to the journal version (equipped with extensions and proofs) of a conference paper published in 2011, actually Brodo, L.

On the expressiveness of the pi-calculus and the mobile ambients
(2011) Lecture Notes in Computer Science
volume 6486 LNCS, pp. 44-59

Line 33 - "a new bio-simulation relation"

> Done.

Line 45 - "looks like"

> Modified following the similar suggestion of the first reviewer.

Page 3

The first 3 items of the contributions are unclear and should be grouped into one item. Instead start each sentence with "we define" or "we show" or something similar.

> Ok, done.

Line 60 - The contribution is unclear (at this point in the paper what are s and s' and what is the relationship among them?) and it also raises the question if it is worth to be considered as a contribution since it is only sketched.

> We have rephrased for clarifying the roles of s and s' ; we have specified that the two last items are not part of the main contributions.

Line 63 - Last contribution is somehow contradictory and the example is unclear at this point in the paper. The contradiction comes from the last sentence in the conclusions section where you say that you are planning to extend the framework to include

communication. The result in section 7.2 is also a sketch, not fully studied, barely a little more than one page.

> Right, we have modified this part of the presentation by separating the actual contributions of the paper from the possible extensions we sketch.

Line 75 - Section 7.2 instead of Section 7

> Done

Line 78 - "SOS operational semantics" -> "structural operational semantics"

> Done

Line 88 - Exactly how many new examples?

> All the examples in Sections 5 and 6.3 are new. This is now clarified in the Introduction.

Page 4

The name of the section 2 should include any of the words background or overview.

> Done

Line 53: Say that S is called the entity set?

Definition 2: I think it would be helpful to state that S is a finite set of entities and $A \subseteq \text{rac}(S)$ is a set of reactions over S .

> Ok, done.

Definition 3: The same as above, say "a finite subset of entities". Overall the definitions need more words to remind the readers what are all the sets about, e.g. a is enabled by W if W includes the reactant set but not the inhibitor set...

> Ok, done.

Definition 4: Replace $n \geq 0$ with n a nonnegative integer; $C_i, D_i \subseteq S$ are set of entities. How is C_0 defined?

> Ok, we implemented the suggestions. Following the standard definition from RSs, we posed no constraints over C_0 that can be whatever set of entities.

Page 6

Line 166: Please give more details of the reason for extending the notion of interactive processes.

> Ok, we have added the sentence "Since we will be able to deal with recursively contexts, we extend the notion of an interactive process to deal with infinite sequences".

Line 173: When you say "we introduce" do you mean "we define"? Is cCNA new as it is not clear at this point? Was it defined in [16]? Are there any paper references for CNA? Be more explicit in the text about the difference between the link-calculus and cCNA (and CNA?).

> cCNA has been introduced in the conference version of this journal submission. We have explained in the introduction and at the beginning of Section 3 the relationship between link/CNA/cCNA, also pointing to the papers where they were originally introduced.

Line 176: The symbol \mathcal{A} is overloaded: it was used earlier for (S,A) a reaction system! Please use another letter or font style for the set of actions.

> Ok.

Line 181: Shouldn't ϵ be defined here?

> Ok.

Give intuition behind a link $\alpha \backslash \beta$: what roles are α and β supposed to have? (input, output, etc.)

> Ok. We added some intuitions.

Page 7

Syntax: the grammar is not complete as it does not define the empty summation, not single $v.P$, nor the operator $+$. Is the operator Sum a summation or a non-deterministic choice? Using the word summation is misleading at this point. It's better to be explicit, so replace the operator Sum with $0 \mid v.P \mid P + Q$.

> Ok. We have clarified these points.

Is the set I finite?

> Yes, we have specified this point.

Line 202: here is the first reference to a paper introducing CNA, it should be also earlier. What is the difference between CNA and link-calculus, it seems that CNA is a subset of link-calculus. Why isn't the name mobility included in CNA and cCNA?

> Ok, we have clarified the origins and differences of link/CNA/cCNA. Mobility is not considered here because it is not necessary to encode RSs and its treatment would add some technical machinery.

Semantics is not completely detailed. Why are rules Rel and Rpar omitted (per Fig.1 caption)? Rel is not even mention in the text on page 7.

> The Rel operator has been removed from the syntax. We have added the Rpar rule. In the syntax we have omitted the relabeling operator, as it is of no use here. It generally serves to reusing components in different situations by only renaming some of the actions, and it is needed to achieve Turing completeness.

Δ is not defined.

> We have added the definition of Δ .

In rule Res shouldn't the condition " $(\nu a)v$ is defined" be included? The same in rule Com for the condition " $v \cdot v'$ defined". Best is to give the full semantics.

> To improve the readability of the rules, definedness of labels is always left implicit. And we have specified it in the main text.

In the Fig.1 caption: SOS semantics -> structural operational semantics.

> Done.

Example 7 on page 7 is not clear at all. What is a forwarder? Or "some sort of forwarder" (this is too colloquial)? For H is it "alternates" or do you mean sequential? What is the dot operator between links in the definition of H ?

> Ok, rephrased.

Page 8

Definition 8: Because you say "vv is also a valid link chain" it implies that v is assumed to be a valid link.

> Ok, we have specified that v is a valid link.

Definition 9: instead of saying "we define the ..." better say "we denote by $\$a\$$ a half left link and by $\$a\$$ a half right link.

> Done.

Definition 10 was difficult to follow. R looks like the set of reactants in a reaction, was that your intention?

> Ok, we changed the description of the example.

You also overload the use of subscript "i" (previously it was denoting a nonnegative integer while here it stands for "input"?) and initially thought that i is a number and the other subscript o is zero. Best to give the intuition explicitly behind the two new subscripts.

> Ok, we have explained the use of the subscripts "i" and "o".

ϵ appears here but it was not defined previously.

> Now it is defined.

The format of the "table" is confusing: please use the booktabs package or default tables and a horizontal line to separate the header from the rest of 3 rows; in the header use capital letters for the initial letter of the first word. Or alternatively, use the definition with 3 choices for the block of chain.

> Ok. we changed the presentation format.

Is R a set or a sequence of names? As on line 236 when you instantiate the expression of the open block with R you only get one block, first a and then b; why not first b and then a?

> Ok, we have changed the formulation of Definition 10.

Line 239: When you say "we present", do you mean "we define" or "we present a new translation" ? It's not clear this is a new.

> Ok, corrected.

Page 9

First line: "we exploit four" -> we exploit 5"

> Done

I suggest you itemize the new 5 names and emphasize their role so that it helps in the rest of the paper identify which one is which.

> We added the itemize format.

What does it mean "progressive number j"?

> We rephrased the sentence.

Page 10:

Introduce r_j , r_{j+1} , p_j , p_{j+1} before use.

> Done.

What does it mean "step n"? If $n=0$ as you mention in parentheses the case of C_0 , then there should be Cxt^0 too, isn't it? This is not clear. Is C_n included in S ? When $e \notin C_n$, does it mean $e \in S \setminus C_n$? What is cxt ?

> Ok, we have corrected and explained the notation.

Move the paragraph before Def11 with the notation conventions before the subsection or paragraph Processes for reactions because you used there reaction names and production names.

> Ok, done.

Is a reaction name a channel name?

> Yes. We reuse reaction names as channel names.

What is a production? Do you mean product?

> Ok. Now it is fixed.

Remind here that A is a subset of reactions in S .

> Ok.

Page 11

The init process is defined recursively?

> Yes, we have specified it in the text.

Page 12

In Example 15 please find another notation for an empty set of inhibitors, why not \emptyset , because it looks like there was a printing error for the second argument in the reactions r_1 and r_2 .

> Ok, we have added the \emptyset symbol.

In the definitions of P_{r_1} and so on, because of the long subscripts the notations appear misleading for each link: the left argument is too far away from the symbol \setminus and that must be due to the fact that you stacked one on top of the other say for example \square and s_{1_i} . Please fix this because it doesn't respect the visual representation of the definition of a link.

> We reduced the spaces.

Page 13

In the middle of the page when you write $\text{flat}(v) = \dots$, you omitted the concatenation symbols $::$. Best would be to remove the $::$ symbol in the definition of the function $\text{flat}(\cdot)$.

> Ok, we have eliminated the $::$ symbol.

Page 14

Please give a reason for omitting the topmost restrictions (as a matter of fact, what are those?).

> Topmost restrictions refer to the series of name restrictions (ν ...) that are put in front of the parallel composition of processes (i.e. at the top of the term). They are useful to force the matching of names in the link chain. When applying rule (Res) all occurrences of the restricted names are transformed in the silent action τ , i.e. restricted names are not visible in the label. If we omit topmost restrictions but we restrict to consider solid chains then we are still guaranteed that all names appearing in the link chain are matched, with the advantage that they appear in the label and we can predicate over them.

In subsection 5.1, what does it mean that a deterministic transition system is coded? Do you mean represented, encoded?

> Ok, we have used “encoded”.

For readers not familiar with RS please explain how is the deterministic LTS in figure 3 represented as the RS at the bottom of the page, how does one get to those reactions?

> Ok, we have added some explanations.

Page 15

Say that $n \in \{1, 2, 3, 4\}$.

> Done.

Page 16

Line 375: text is missing.

> Fixed.

Page 17

In the definition of $P(T, \tilde{T})$ why is there a space after the + sign? The same questions for the definitions of P_F and \bar{P}_F .

> Fixed.

Page 18

Line 396: "only considers"

> Done.

Line 397: "when they expose"

> Done.

Line 412: "the mentioned" -> "fore mentioned"

> Done.

The justification that "the classical bisimulation seems to be inappropriate" is weak, please elaborate.

> Ok, we have better explained this point.

"A high number of interactions occur every seconds". It should be second, but the question is more about the chosen time unit.

> Ok, we have used a more generic “computational stage”.

Page 19:

"to only the limited events of interest" -> "to only a limited set of event of interest".

> Done.

How do you chose this set?

> We have explained this point.

What is a "bisimulation game"?

> We have removed the expression "bisimulation game", as it has no relevance here.

Anyway, the bisimulation game can be seen as a game between an attacker (who wants to disproved the equivalences of processes) and a defender (who wants to prove the equivalence).

Line 436: " s_i blocked" -> " s_i been blocked"

> Done.

What are "simple strings" as compared to more complex strings?

> Ok, we have removed "simple".

The use of the same font style for syms, decs, reacts, prods, and cxt is misleading, as syms, decs, reacts and prods are sets, while cxt is an element of a set.

> Ok, we have changed the presentation.

Definition 21: α is overloaded, before it denoted channels in the set $\{a,b,c,\dots\}$

> Ok, we have changed α with η .

Page 20

When talking about the singleton, that's not "a string composed by a single symbol ...", best to say that it's either a symbol α or the wildcard ? ...

> Ok, we have changed the word " singleton" with "atomic assertion".

The next sentence also needs to be changed to remove "strings composed by" -- that's not correct in English.

> Done.

Replace "We write \star as an abbreviation of $?^*$." with "We denote by

> Ok, done.

Definition 22: what does the symbol after \subseteqq mean?

> That symbol denotes a powerset, as we have specified in the text.

Example 24 first line:

formulas -> assertions

> Done.

Is there a typo in the third item: p_j or r_j ?

> It is p_j , as it marks the part of the link chain where the products of the reaction r_j is specified.

Page 21

"behaviour of the contest" -- what contest is that?

> Ok, it is context. We fixed it.

"parametric to the assertion" -> "parametric w.r.t. the assertion" ?

> Done.

Introduce the abbreviation HML on line 513.

> Done.

Page 22

Line 519: "the set of process" -> "the set of cCNA processes"

> Done.

Line 526: make this paragraph a definition and emphasise "bio-logically equivalent w.r.t. F"

> Done.

"the exactly the same" -> "exactly the same"

> Done.

Page 23

Line 535: it is not obvious how to go from a RS to a LTS, please detail.

> Ok, we have added some explanations in the text.

Since in section 7 you sketch some ideas, shouldn't that be helpful to say it in the section title? Maybe "Towards ..." or "Enhancing ..." or simply "Future Work"?

> Ok, done.

Page 24

Table 1 is unclear. Please add headers or just remove it completely and describe the two RSs in text. Again the missing second argument in the reaction rules is misleading as if it's a latex/printing error. Why not \emptyset ?

> Ok, we have deleted the Table 1, and explained the two RSs in the text.

"to suite this" -> "to suit this"

> Done.

The translation with several "by definition" symbols (triangle on top of the equal sign) is misleading, please separate on both pages 24 and 25.

> Ok. Fixed.

Page 25

It's only in the first line of the conclusions you are clearly stating that cCNA is defined in this paper.

> We have rephrased the whole paragraph.

"link chains and no more single links ..." -> "link chains generalizing the original single links. This was described initially in the ..."

> Ok.

"This variant ..." -> "cCNA"

> Ok.

"a faithful embedding of RSs" -> a faithful embedding into what? Best to rephrase this sentence.

> Ok, we have rephrased the sentence.

Enumerate clearly the benefits: "This translation ..." -> "This translation shows several benefits: 1) ... , 2)... , 3)... , and 4)

"We have then modified ..." -> "We defined a new bisimulation relation, bio-simulation, based on ..."

"We defined a new assertion language for specifying properties of RS encodings in cCNA and showed how this language extends ..."

> Ok, we have rephrased the whole paragraph.

The sentence "We believe ..." is vague in relation to the three cited papers, could you give more details?

The sentence "We are confident ..." is again vague and doesn't bring anything here.

>Ok, we have rephrased the sentences.

"We can also exploit the implementation" -> "and we will make use of the implementation [26] ... "

> Ok, we have rephrased the sentence.

The last sentence basically says what you did already in Section 7.2. Is there more to it? If so, please expand.

> Ok, we have removed the sentence.

In the bibliographic references, paper [16] has a different title on the Springer website. Please check.

> Done.

A process algebraic approach to reaction systems ^{*}

Linda Brodo^b, Roberto Bruni^a, Moreno Falaschi^c

^a*Dipartimento di Informatica, Università di Pisa, Italy*

^b*Dipartimento di Scienze Economiche e Aziendali, Università di Sassari, Italy*

^c*Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, Univ. di Siena, Italy*

Abstract

In the area of Natural Computing, reaction systems are a qualitative abstraction inspired by the functioning of living cells, suitable to model the main mechanisms of biochemical reactions. This model has already been applied and extended successfully to various areas of research. Reaction systems interact with the environment represented by the context, and pose challenges for further extensions as well as for the implementation, as it is a new computation model. In this paper we consider a variant of the `link`-calculus, which allows to model multiparty interaction in concurrent systems, and show that we can represent reaction systems as processes in a modular way, by representing the behaviour of each entity and preserving faithfully their features. We show the correctness and completeness of our embedding. We illustrate our framework by showing the embedding of a few examples expressing computer science and biological applications. On top of the LTS semantics for reaction systems provided by our framework, we then adapt the classical notion (in concurrency) of bisimulation to make it more suitable for studying properties of reaction systems. In particular, we define a new assertion language based on regular expressions, which allows to specify the properties of interest, and use it to extend Hennessy-Milner logic to our framework. Interestingly, the novel notion of bisimilarity and logical equivalence that are defined parametrically on some assertion of interest are proved to coincide, like in the classical case. Finally, our methodology can contribute to increase the expressiveness of reaction systems, by exploiting the interaction among different reaction systems.

Keywords: process algebras, reaction systems, assertion language, HM-logic

1. Introduction

Natural Computing is an emerging area of research which has two main aspects: human designed computing inspired by nature, and computation per-

^{*}Research partially supported by Università degli Studi di Sassari *Fondi di Ateneo per la ricerca 2019*.

Email addresses: `brodo@uniss.it` (Linda Brodo), `bruni@di.unipi.it` (Roberto Bruni), `moreno.falaschi@unisi.it` (Moreno Falaschi)

formed in nature. Reaction Systems (RSs) [1] are a rewriting formalism inspired by the way biochemical reactions take place in living cells. This theory has already shown to be relevant in several different fields, such as computer science [2], biology [3, 4, 5, 6], molecular chemistry [7]. Reaction Systems formalise the mechanisms of biochemical systems, such as *facilitation* and *inhibition*. As a qualitative approximation of the real biochemical reactions, they consider if a necessary reagent is or not present, and likewise they consider if an inhibiting molecule is or not present. The possible reactants and inhibitors are called ‘entities’. RSs model in a direct way the interaction of a living cell with the environment (called ‘context’). However, two RSs are seen as independent models and do not interact.

In this paper, we present an encoding from RSs, to the open multiparty process algebra cCNA,¹ a variant of the `link`-calculus [8, 9] without name mobility. This formalism allows several processes to synchronise and communicate altogether, at the same time, with a new interaction mechanism based on links and link chains. The initial motivation for introducing this mechanism was to encode Mobile Ambients [10], getting a much stronger operational correspondence than any available in the literature, such as the one in [11]. Later it was shown that the `link`-calculus allowed one to easily encode calculi for biology equipped with membranes, as in [12].

We illustrate our embedding by means of some examples coming both from the computer science and the biological field. We also show that our embedding preserves the main features of RSs, and prove its correctness and completeness from the operational semantics viewpoint.

Then, we present a methodology for verifying formally properties of Reaction Systems. The classical notion of bisimulation for process algebras allows to consider two processes as equivalent when one process can simulate all the actions executed by the other one and vice versa. In this paper we define a new notion of bisimilarity which takes into account the characteristics of biological systems. We define a new *bio-simulation* having in mind the possibility that two interacting systems may be compared w.r.t. a subset of the possible biological actions. This is useful for concentrating on the sub-model that one may need to consider for a specific study or application, without getting lost in the complexity of the full biological system, or network. The notion of bio-simulation relies on a new and simple assertion language, which allows to focus on some properties of the Reaction Systems to be verified. In fact, bio-similarity is parametric on a given assertion of interest. Then we prove that the well-known logical characterisation of bisimilarity in terms of Hennessy-Milner Logic [13] (HML) can be extended to the case of bio-similarity by tailoring HML formulas to the same assertion of interest used in bio-similarity. As HML is a powerful formalism for specifying properties of labeled transition systems, its variant introduced here looks a suitable option to specify formulas over complex labels by abstracting from unnecessary details.

¹After ‘chained Core Network Algebra’.

Our main contributions are as follows:

- the behaviour of the context, for each single entity, can be specified in a recursive way, as ordinary processes;
- along the same lines, non-deterministic contexts can be seamlessly expressed in our setting;
- when deterministic contexts are considered, the cCNA computation is deterministic and mimics the evolution of the corresponding RS;
- we define an assertion language to specify local properties of Reaction Systems and exploit it to define a novel notion of behavioural equivalence for Reaction Systems, called bio-similarity;
- we show that our assertions can be used to extend the Hennessy-Milner logic to our framework, in such a way that logical equivalence of processes coincide with bio-similarity;
- we sketch how one can express the behaviour of entity mutation, in such a way that the mutated entity s' can take part to only a subset of rules requiring entity s ;
- we show that with a little coding effort, our encoding allows two RSs to communicate; i.e. we can model scenarios where a subset of those entities that the context can provide, are instead provided by a second RS.

The main drawback of our proposal, is that the cCNA translation is verbose. Nevertheless it is clear that our translation can be automatised by means of a proper front-end in an implementation of the `link`-calculus. The examples that we propose also show that some optimisations are possible to reduce the coding when suitable assumptions are made about the provision of entities.

As we have remarked, in our translation, Reaction Systems get the ability to interact between them in a synchronized manner. This interaction is not foreseen in the basic RS framework, as it can only happen with the context. By exploiting recursion, the kind of interactions which can be defined can be complex and expressive. Example 33 and more in general the discussion in Section 7 show that the interaction between RSs can help to model new scenarios.

Related work. Process calculi have been used successfully to model biological processes, see [14] for a recent survey. We are not aware of any SOS operational semantics for the RS; it seems not the case that a deterministic behaviour, as the one of the RS, once the initial state is fixed, could be defined by inference rules that classically are applied to define non-deterministic transition systems. The reversible computation paradigm for RS extends the RS framework by allowing backward computations as well as forward computations; for this goal a set of inference rules for rewriting logic has been defined in [15] to exactly keep trace of those elements that dissolve in the next computation step.

A preliminary version of this paper appeared in [16]. There are several major differences w.r.t. the paper [16], which is here extended as follows:

- we present several new examples to illustrate our framework, and give more detailed explanations about its use;
- we introduce an assertion language to specify the properties of Reaction Systems;
- we define the notion of bio-simulation to relate Reaction Systems and compare their behaviours;
- we show that our assertion language allows to specify properties extending to our framework the Hennessy-Milner logic;
- we include here all proofs of main results.

Structure of the paper. Section 2 describes RSs and their semantics (interactive processes). Section 3 describes briefly the cCNA process algebra and its operational semantics. Section 4 defines the embedding of RSs in cCNA processes and shows some simple examples to illustrate it. Section 5 presents a couple of examples with computer science and biological applications. Section 6 presents a methodology for the formal verification of properties of Reaction Systems that are expressed in a novel assertion language. Section 7 presents some features and advantages of our embedding for the compositionality of RSs. Finally, Section 8 discusses future work, and concludes.

2. Reaction Systems

Natural Computing is concerned with human-designed computing inspired by nature as well as with computation taking place in nature. The theory of Reaction Systems [1] was born in the field of Natural Computing to model the behaviour of biochemical reactions taking place in living cells. Despite its initial aim, this formalism has shown to be quite useful not only for modeling biological phenomena, but also for the contributions which is giving to computer science [2], theory of computing, mathematics, biology [3, 4, 5, 6], and molecular chemistry [7]. Here we briefly review the basic notions of RSs, see [1] for more details.

The mechanisms that are at the basis of biochemical reactions and thus regulate the functioning of a living cell, are *facilitation* and *inhibition*. These mechanisms are reflected in the basic definitions of Reaction Systems.

Definition 1 (Reaction). A reaction is a triplet $a = (R, I, P)$, where R, I, P are finite, non empty sets and $R \cap I = \emptyset$. If S is a set such that $R, I, P \subseteq S$, then a is a reaction in S .

The sets R, I, P are also written R_a, I_a, P_a and called the *reactant set* of a , the *inhibitor set* of a , and the *product set* of a , respectively. All reactants are needed for the reaction to take place. Any inhibitor blocks the reaction if it is present. Products are the outcome of the reaction. Also, $R_a \cup I_a$ is the set of the

resources of a and $rac(S)$ denotes the set of all reactions in S . Because R and I are non empty, all products are produced from at least one reactant and every reaction can be inhibited in some way. Sometimes artificial inhibitors are used that are never produced by any reaction. For the sake of simplicity, in some examples, we will allow I to be empty.

Definition 2 (Reaction System). A Reaction System (RS) is an ordered pair $\mathcal{A} = (S, A)$ such that S is a finite set, and $A \subseteq rac(S)$.

The set S is called the *background set* of \mathcal{A} , its elements are called *entities*, they represent molecular substances (e.g., atoms, ions, molecules) that may be present in the states of a biochemical system. The set A is the set of *reactions* of \mathcal{A} . Since S is finite, so is A : we denote by $|A|$ the number of reactions in A .

Definition 3 (Reaction Result). Let W be a finite subset of S .

1. Let a be a reaction in S . Then a is enabled by W , denoted by $en_a(W)$, if $R_a \subseteq W$ and $I_a \cap W = \emptyset$. The result of a on W , denoted by $res_a(W)$, is defined by: $res_a(W) = P_a$, if $en_a(W)$, and $res_a(W) = \emptyset$ otherwise.
2. Let A be a finite set of reactions. The result of A on W , denoted by $res_A(W)$, is defined by: $res_A(W) = \bigcup_{a \in A} res_a(W)$.

The theory of Reaction Systems is based on the following assumptions.

- **No permanency.** An entity of a set W vanishes unless it is sustained by a reaction. This reflects the fact that a living cell would die for lack of energy, without chemical reactions.
- **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 taken into account.
- **Threshold nature of resources.** From the previous item, we assume that either an entity is available and there is enough of it (i.e. there are no conflicts), or it is not available at all.

The dynamic behaviour of a RS is formalized in terms of *interactive processes*.

Definition 4 (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 $\pi = (\gamma, \delta)$ of finite sequences s.t. $\gamma = \{C_i\}_{i \in [0, n]}$ and $\delta = \{D_i\}_{i \in [0, n]}$ where $C_i, D_i \subseteq S$ for any $i \in [0, n]$, $D_0 = \emptyset$, and $D_i = res_A(D_{i-1} \cup C_{i-1})$ for any $i \in [1, n]$.

Living cells are seen as open systems that continuously react with the external environment, in discrete steps. The sequence γ is the *context sequence* of π and represents the influence of the environment on the Reaction System. The sequence δ is the *result sequence* of π and it is entirely determined by γ and A . The sequence $\tau = W_0, \dots, W_n$ with $W_i = C_i \cup D_i$, for any $i \in [0, n]$ is called a

state sequence. Each state W_i in a state sequence is the union of two sets: the context C_i at step i and the result $D_i = \text{res}_A(W_{i-1})$ from the previous step.

For technical reasons, we extend the notion of an interactive process to deal with infinite sequences.

Definition 5 (Extended Interactive Process). Let $\mathcal{A} = (S, A)$ be a RS, and let $\pi = (\gamma, \delta)$ be an n -step interactive process, with $\gamma = \{C_i\}_{i \in [0, n]}$ and $\delta = \{D_i\}_{i \in [0, n]}$. Then, we let $\pi^\infty = (\gamma^\infty, \delta^\infty)$ be the extended interactive process of π , defined as $\gamma^\infty = \{C'_i\}_{i \in \mathbb{N}}$, $\delta^\infty = \{D'_i\}_{i \in \mathbb{N}}$, where:

$$C'_j = \begin{cases} C_j & \text{if } j \in [0, n] \\ \emptyset & \text{if } j > n \end{cases} \quad D'_j = \begin{cases} D_0 & \text{if } j = 0 \\ \text{res}_A(D'_{j-1} \cup C'_{j-1}) & \text{if } (j \geq 1) \end{cases}$$

Given an extended interactive process $\pi = (\gamma, \delta)$, we denote by π^k the shift of π starting at the k -th state sequence; formally we let $\pi^k = (\gamma^k, \delta^k)$ with $\gamma^k = \{C'_i\}_{i \in \mathbb{N}}$, $\delta^k = \{D'_i\}_{i \in \mathbb{N}}$ with $C'_0 = C_k \cup D_k$, $D'_0 = \emptyset$, and $C'_i = C_{i+k}$, $D'_i = D_{i+k}$ for any $i \geq 1$.

3. Chained CNA (cCNA)

In this section we introduce the syntax and operational semantics of a variant of the link-calculus [8], the cCNA (chained CNA) where the prefixes are link chains and not just links.

Link Chains. Let \mathcal{C} be the set of channels, ranged over by a, b, \dots , and let $\mathcal{A} = \mathcal{C} \cup \{\tau\} \cup \{\square\}$ be the set of actions, ranged over by α, β, \dots , where the symbol τ denotes a *silent* action, while the symbol \square denotes a *virtual* (non-specified) action. A *link* is a pair $\ell = \alpha \setminus \beta$; it is *solid* if $\alpha, \beta \neq \square$; the link $\square \setminus \square$ is called *virtual*. A link is *valid* if it is solid or virtual. We let \mathcal{L} be the set of valid links. A *link chain* is a finite sequence $v = \ell_1 \dots \ell_n$ of (valid) links $\ell_i = \alpha_i \setminus \beta_i$ such that:

1. for any $i \in [1, n-1]$, $\begin{cases} \beta_i, \alpha_{i+1} \in \mathcal{C} & \text{implies } \beta_i = \alpha_{i+1} \\ \beta_i = \tau & \text{iff } \alpha_{i+1} = \tau \end{cases}$
2. $\exists i \in [1, n]. \ell_i \neq \square \setminus \square$.

Virtual links represent missing elements of a chain. A chain is called *solid* if it does not contain any virtual link. The equivalence \blacktriangleleft models expansion/contraction of virtual links to adjust the length of a link chain.

Definition 6 (Equivalence \blacktriangleleft). We let \blacktriangleleft be the least equivalence relation over link chains closed under the axioms (whenever both sides are well defined):

$$\begin{array}{lll} v \square \setminus \square & \blacktriangleleft & v \\ \square \setminus \square v & \blacktriangleleft & v \\ v_1 \square \setminus \square \setminus \square v_2 & \blacktriangleleft & v_1 \square \setminus \square v_2 \\ v_1 \alpha \setminus \alpha \setminus \beta v_2 & \blacktriangleleft & v_1 \alpha \setminus \alpha \setminus \beta v_2 \end{array}$$

Two link chains of equal length can be merged whenever each position occupied by a solid link in one chain is occupied by a virtual link in the other chain and solid links in adjacent positions match. Positions occupied by virtual

links in both chains remain virtual. Merging is denoted by $v_1 \bullet v_2$. For example, given $v_1 = \tau \backslash_a \square \backslash_b \square$, $v_2 = \square \backslash_a \square \backslash_b \square$ and $v = \tau \backslash_a \square \backslash_b \square$ we have $v_1 \bullet v_2 = v$, whereas $v_1 \bullet v$ is not defined. Notably the merge operation is commutative and associative.

Some names in a link chain can be restricted as non observable and transformed into silent actions τ . This is possible only if they are matched by some adjacent link. Restriction is denoted by $(\nu a)v$. For example, given $v = \tau \backslash_a \square \backslash_b \square$ as above, we have $(\nu a)v = \tau \backslash_\tau \square \backslash_b \square$, whereas $(\nu b)v$ is not defined.

Syntax. The set of cCNA processes, denoted as \mathcal{P} and ranged over by P, Q , is defined by the following grammar:

$$P, Q ::= \sum_{i \in I} v_i.P_i \mid P|Q \mid (\nu a)P \mid P[\phi] \mid A$$

where v_i is a link chain, ϕ is a channel renaming function, and A is a process identifier for which we assume a definition $A \triangleq P$ is available in a given set Δ of (possibly recursive) process definitions. We let $\mathbf{0}$, the inactive process, denote the empty summation.

The syntax of cCNA extends that of CNA [9] by allowing to use link chains as prefixes instead of links. For the rest it features nondeterministic (guarded) choice, parallel composition, restriction, relabelling and possibly recursive definitions. Here we do not consider name mobility, which is present instead in the link-calculus.

Semantics. The operational semantics of cCNA is defined in the SOS style by the inference rules in Fig.1. The rules are reminiscent of those for Milner's CCS and they essentially coincide with those of CNA in [9]. The only difference is due to the presence of prefixes that are link chains. Briefly: rule *(Sum)* selects one alternative and puts as label a possible contraction/expansion of the link chain in the selected prefix; rule *(Ide)* selects one transition of the defining process for a constant; rule *(Res)* restricts some names in the label (it cannot be applied when $(\nu a)v$ is not defined); rules *(Lpar)* and *(Rpar)* account for interleaving in parallel composition; rule *(Com)* synchronises interactions (it cannot be applied when $v \bullet v'$ is not defined).

Example 7. As some simple examples, consider the recursive process definitions $H \triangleq a \backslash_b . a \backslash_c . H$ and $K \triangleq a \backslash_b . K + a \backslash_c . K$: the former is some sort of forwarder that alternates between providing a link from a to b and to c ; the latter is a nondeterministic forwarder that at each step can link a to b or to c .

Analogously to CNA, the operational semantics of cCNA satisfies the so called Accordion Lemma: whenever $P \xrightarrow{v} P'$ and $v' \blacktriangleleft v$ then $P \xrightarrow{v'} P'$.

3.1. Notation for link chains

Hereafter we make use of some new notations for link chains that will facilitate the presentation of our translation.

$$\begin{array}{c}
\frac{v \blacktriangleright v_j \quad j \in I}{\sum_{i \in I} v_i.P_i \xrightarrow{v} P_j} (Sum) \quad \frac{P \xrightarrow{v} P' \quad (A \triangleq P) \in \Delta}{A \xrightarrow{v} P'} (Ide) \\
\\
\frac{P \xrightarrow{v} P'}{(\nu a)P \xrightarrow{(\nu a)v} (\nu a)P'} (Res) \quad \frac{P \xrightarrow{v} P'}{P|Q \xrightarrow{v} P'|Q} (Lpar) \\
\\
\frac{P \xrightarrow{v'} P' \quad Q \xrightarrow{v} Q'}{P|Q \xrightarrow{v \bullet v'} P'|Q'} (Com)
\end{array}$$

Figure 1: SOS semantics of the cCNA (rules (Rel) and $(Rpar)$ omitted).

Definition 8 (Replication). Let v be a link chain such that vv is also a valid link chain. Its n times replication v^n is defined recursively by letting $v^0 = \epsilon$ (i.e. the empty chain) and $v^n = vv^{n-1}$.

For example, the expression $(a \setminus_b^{\square} \setminus_{\square}^{\square})^3$ denotes the chain $a \setminus_b^{\square} \setminus_{\square}^{\square} a \setminus_b^{\square} \setminus_{\square}^{\square} a \setminus_b^{\square} \setminus_{\square}^{\square}$.
 230 Instead the expression $(a \setminus_b)^2$ is ill-defined because a does not match with b .

Then, we introduce the notation for *half links* that will be used in conjunction with the *open block of chain* to form regular link chains.

Definition 9 (Half links). Let a be a channel name, we define the *half left link* $a \setminus$, and the *half right link* \setminus_a .

Definition 10 (Open block). Let R be a set of names. We define an *open block* as $(\setminus_{a \in R}^{\square} \setminus_{\square}^{a_o})$, where a_i and a_o are annotated version of the name a , as

set	block of chain	result
$R = \emptyset$	$(\setminus_{a \in R}^{\square} \setminus_{\square}^{a_o})$	ϵ
$R = \{b\}$	$(\setminus_{a \in R}^{\square} \setminus_{\square}^{a_o})$	$\square \setminus_{b_i}^{b_o}$
$R = \{b\} \cup R'$	$(\setminus_{a \in R}^{\square} \setminus_{\square}^{a_o})$	$\square \setminus_{b_i}^{b_o} \setminus (\setminus_{a \in R'}^{\square} \setminus_{\square}^{a_o})$

235 We then combine half links and open blocks to form regular link chains. For example, for $R = \{a, b\}$ the expression $(\setminus_{c \in R}^{\square} \setminus_{\square}^{c_o})$ denotes the block $\square \setminus_{a_i}^{a_o} \setminus_{b_i}^{b_o}$; and the expression $r_1 \setminus (\setminus_{c \in R}^{\square} \setminus_{\square}^{c_o}) \setminus_{r_2}$ denotes the chain $r_1 \setminus_{a_i}^{\square} \setminus_{b_i}^{a_o} \setminus_{b_i}^{b_o} \setminus_{r_2}$.

4. From Reaction Systems to cCNA

240 Here we present a translation from Reaction Systems to cCNA. The idea is to define separated processes for representing the behaviour of each entity, each reaction, and for the provisioning of each entity by the context.

Processes for entities. Given an entity $s \in S$, we exploit four different names for the interactions over s : names s_i, s_o are used to test the presence of s in the system; names $\widehat{s}_i, \widehat{s}_o$ are used to test the provisioning of s from the context; names $\widetilde{s}_i, \widetilde{s}_o$ are used to test the production of s by some reaction; names $\overline{s}_i, \overline{s}_o$ are used to test the absence of s in the system; and names $\underline{s}_i, \underline{s}_o$ are used to test the absence of s from the context. We let P_s be the process implementing the presence of s in the system, and \overline{P}_s be the one for its absence. They can be seen as instances of the same template, which is given below.

$$\begin{aligned}
P_s &\triangleq E(s, \widetilde{s}, \widehat{s}, \underline{s}) & \overline{P}_s &\triangleq E(\overline{s}, \widetilde{s}, \widehat{s}, \underline{s}) \\
E(s, \widetilde{s}, \widehat{s}, \underline{s}) &\triangleq \sum_{h,k \geq 0} (s_i \backslash_{s_o} \square)^h \widehat{s}_i \backslash_{\widehat{s}_o} \square (\widetilde{s}_i \backslash_{\widetilde{s}_o} \square)^k . P_s \\
&\quad + \sum_{h \geq 0, k \geq 1} (s_i \backslash_{s_o} \square)^h \underline{s}_i \backslash_{\underline{s}_o} \square (\widetilde{s}_i \backslash_{\widetilde{s}_o} \square)^k . P_s \\
&\quad + \sum_{h \geq 0} (s_i \backslash_{s_o} \square)^h \underline{s}_i \backslash_{\underline{s}_o} . \overline{P}_s
\end{aligned}$$

The first line of $E(s, \widetilde{s}, \widehat{s}, \underline{s})$ accounts for the case where s is tested for presence by h reactions and produced by k reactions, while being provided by the context ($\widehat{s}_i \backslash_{\widehat{s}_o}$). Thus, s will be present at the next step (the continuation is P_s). Here h and k are not known a priori and therefore any combination is possible. In practice, by knowing the number of reactions that test s , we can bound the maximum values of h and k . The second line accounts for the analogous case where s is not provided by the context ($\underline{s}_i \backslash_{\underline{s}_o}$). The condition $k \geq 1$ guarantees that s will remain present (the continuation is P_s). The third line accounts for the case where s is tested for presence, but it is neither produced nor provided by the context. Therefore, in the next step s will be absent in the system (the continuation is \overline{P}_s). Note that in the case of \overline{P}_s the test for presence of s in the system is just replaced by the test for its absence.

Processes for reactions. We assume that each reaction a is assigned a progressive number j . The process for reaction $aj = (R_j, I_j, P_j)$ must assert either the possibility to apply the reaction or its impossibility. The first case happens when all its reactants are present (the link $s_i \backslash_{s_o}$ is requested for any $s \in R_j$) and all its inhibitors are absent (the link $\overline{e}_i \backslash_{\overline{e}_o}$ is requested for any $e \in I_j$), then the product set is released (the link $\widetilde{c}_i \backslash_{\widetilde{c}_o}$ is requested for any $c \in P_j$). The next case can happen for two reasons: one of the reactants is absent (the link $\overline{s}_i \backslash_{\overline{s}_o}$ is requested for some $s \in R_j$) or one of the inhibitors is present (the link $e_i \backslash_{e_o}$ is requested for some $e \in I_j$). The process is recursive so that reactions can be

applied at any step.

$$\begin{aligned}
P_{aj} &\triangleq \\
&r_j \setminus \left(\bigsqcup_{s \in R_j} \square \setminus \underline{s}_o \right) \setminus \left(\bigsqcup_{e \in I_j} \square \setminus \bar{e}_o \right) \setminus_{r_{j+1}} \square \setminus_{p_j} \left(\bigsqcup_{c \in P_j} \square \setminus \tilde{c}_o \right) \setminus_{p_{j+1}} . P_{aj} \quad \{aj \text{ is applicable}\} \\
&+ \\
&\sum_{s \in R_j} r_j \setminus \square \setminus \bar{s}_o \setminus_{r_{j+1}} \square \setminus_{p_j} \setminus_{p_{j+1}} . P_{aj} \quad \{aj \text{ is not applicable}\} \\
&+ \\
&\sum_{e \in I_j} r_j \setminus \square \setminus e_o \setminus_{r_{j+1}} \square \setminus_{p_j} \setminus_{p_{j+1}} . P_{aj} \quad \{aj \text{ is not applicable}\}
\end{aligned}$$

We exploit names r_j, p_j to join the chains provided by the application of all the reactions. Channels r_j and r_{j+1} enclose the enabling/disabling condition of reaction aj . Channels p_i and p_{j+1} enclose the links related to the entities produced by aj . We will see that all the link chain labels of transitions follow the same schema: first we find all the reactions limited to the reactants and inhibitors (chained using r_j channels), then all the supplies by the contexts (chained using the channel cxt , to be introduced next), and finally the products for all the reactions (chained using p_j channels). In the following there is an example explaining this schema.

Processes for contexts. The context can vary at each step n and for each entity $s \in S$, the context must say if the entity is provided or not. Let us denote by C_n the context at step n , i.e. the set of entities that are made available. Correspondingly, we introduce another process Cxt^n defined as follows:

$$Cxt^n \triangleq cxt \setminus \left(\bigsqcup_{s \in C_n} \square \setminus \underline{s}_o \right) \setminus \left(\bigsqcup_{e \notin C_n} \square \setminus \bar{e}_o \right) \setminus_{p_1} . Cxt^{n+1}$$

We only consider Cxt^n with $n > 0$, as the entities that are present at step zero are considered to be present in the initial system (if $s \in C_0$ the process P_s will be present initially, otherwise \bar{P}_s will be present).

In the following we use the following conventions for denoting different categories of names:

- $decs \triangleq \{s, \bar{s}, \tilde{s}, \hat{s} \mid s \in S\}$ is the set of names for decorated entities (without subscripts i and o);
- $ents \triangleq \{d_i, d_o \mid d \in decs\}$ is the set names for entities;
- $reacts \triangleq \{r_1, \dots, r_{|A|}\}$ is the set of reaction names r_j associated with each reaction aj ;
- $prods \triangleq \{p_1, \dots, p_{|A|}\}$ is the set of names for production p_j , associated with reaction aj .

Definition 11 (Translation). Let $\mathcal{A} = (S, A)$ be a RS, and let $\pi = (\gamma, \delta)$ be an extended interactive process in \mathcal{A} , with $\gamma = \{C_i\}_{i \in \mathbb{N}}$. We define its cNA translation $\llbracket \mathcal{A}, \gamma \rrbracket$ as follows:

$$\llbracket \mathcal{A}, \gamma \rrbracket \triangleq (\nu \text{ names}) \left(I \mid \prod_{a \in A} P_a \mid Cxt^1 \mid \prod_{s \in C_0} P_s \mid \prod_{s \notin C_0} \bar{P}_s \right)$$

where $names = reacts \cup ents \cup prods \cup \{cxt\}$. For technical reasons, we introduce the init process $I \triangleq \tau \backslash_{r_1}.I$ to allow the name r_1 to be matched at the start of any chain; also, for notational convenience, we fix that $r_{u+1} = cxt$ and $p_{u+1} = \tau$, for $u = |A|$.

It is important to observe that, for each transition, our cCNA encoding requires all the processes running in parallel to interact in that transition. This is due to the fact that all the channels r_j, p_j, cxt , including those for decorated names $s_i, s_o, \bar{s}_i, \bar{s}_o$ are restricted. Each reaction defines a pattern to be satisfied, i.e. each reaction inserts as many virtual links as the number of reactants, inhibitors, and products, as required by the corresponding reaction.

Lemma 12. *Let $\mathcal{A} = (S, A)$ be a RS and let $\pi = (\gamma, \delta)$ be an extended interactive process in \mathcal{A} . Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ its cCNA translation. If exists P' such that $P \xrightarrow{(\nu names)v} P'$ is a transition of P , then*

1. *for each reaction $aj \in A$, the corresponding channels r_j and p_j appear in v ; for each entity $s \in S$, the corresponding channel s (suitably decorated) appear in v ; the channel cxt appears in v ;*
2. *for each reaction $aj \in A$ and each entity $s \in S$, each virtual link offered by processes P_a and Cxt is overlapped by exactly one solid link offered by processes representing entities.*

The topmost restriction $(\nu names)$ appearing in the process $\llbracket \mathcal{A}, \gamma \rrbracket$ serves to guarantee that all names appearing in a link of the chain labelling a transition are matched. Since all names appearing in any prefix of $\llbracket \mathcal{A}, \gamma \rrbracket$ are restricted, in the transition $\llbracket \mathcal{A}, \gamma \rrbracket \xrightarrow{(\nu names)v} P'$ it means that the observation $(\nu names)v$ has the form $\tau \backslash_{\tau} \dots \tau \backslash_{\tau}$, i.e., it is silent, and that v is solid. Later on we will be interested in reasoning about the actual chain v used in the transition. It has the peculiarity to start and end with silent actions and to include all names in $reacts \cup prods \cup \{cxt\}$. As a matter of notation we call such chain v *complete*.

Definition 13 (Complete Chain). A chain v is called *complete* if it is solid (i.e., it contains no virtual link) and has silent actions τ at its extremes. We write $P \xRightarrow{v} P'$ to mean that $P \xrightarrow{v} P'$ with v complete.

We will then use $\langle \mathcal{A}, \gamma \rangle$ to refer to the encoding without topmost name restrictions, i.e.,

$$\langle \mathcal{A}, \gamma \rangle \triangleq I \mid \prod_{a \in A} P_a \mid Cxt^1 \mid \prod_{s \in C_0} P_s \mid \prod_{s \notin C_0} \bar{P}_s.$$

and we will focus on the complete transitions of $\langle \mathcal{A}, \gamma \rangle$.

The following Corollary immediately follows from Lemma 12.

Corollary 14. *Let $\mathcal{A} = (S, A)$ be a RS and let $\pi = (\gamma, \delta)$ be an extended interactive process in \mathcal{A} . Let $Q = \langle \mathcal{A}, \gamma \rangle$ and $P = \llbracket \mathcal{A}, \gamma \rrbracket = (\nu names)Q$. Then $P \xrightarrow{(\nu names)v} P'$ iff $Q \xRightarrow{v} Q'$ and $P' = (\nu names)Q'$.*

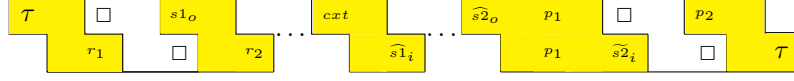


Figure 2: The link chain structure arising from reactions and context processes.

Example 15. Let \mathcal{A} be a RS whose specification contains two entities, $s1$ and $s2$, and the reactions $r_1 = (s1, \cdot, s2)$ and $r_2 = (s2, \cdot, s1)$ that produce $s2$ if $s1$ is present and $s1$ if $s2$ is present. For simplicity we do not consider inhibitors, but the reader can assume a void inhibitor is present in both reactions. Then, we assume an extended interactive process $\pi = (\gamma, \delta)$ where the context γ provides $s1$ and $s2$ at every step, but we assume that only $s1$ is initially present. The corresponding cCNA process is $\llbracket \mathcal{A}, \gamma \rrbracket \triangleq (\nu \text{ names}) \langle \mathcal{A}, \gamma \rangle$, with

$$\langle \mathcal{A}, \gamma \rangle \triangleq I \mid P_{s1} \mid \bar{P}_{s2} \mid P_{r1} \mid P_{r2} \mid Cxt$$

where:

$$\begin{aligned} P_{r1} &\triangleq r_1 \setminus s1_i \setminus s1_o \setminus r_2 \setminus s2_i \setminus p1 \setminus s2_o \setminus p2 \cdot P_{r1} + \dots; \\ P_{r2} &\triangleq \dots + r_2 \setminus s2_i \setminus s2_o \setminus cxt \setminus p2 \cdot P_{r2} + \dots; \\ P_{s1} &\triangleq s1_i \setminus s1_o \setminus s1_i \cdot P_{s1} + \dots; \\ \bar{P}_{s2} &\triangleq s2_i \setminus s2_o \setminus s2_i \setminus s2_o \setminus s2_i \cdot P_{s2} + \dots; \\ Cxt &\triangleq cxt \setminus s1_i \setminus s1_o \setminus s2_i \setminus s2_o \setminus p1 \cdot Cxt \end{aligned}$$

For clarity of exposition, we show the code of the processes just in part, to focus on the prefixes that will be involved in the first transition of the system. In Figure 2 we show the structure of a link chain label related to the execution of such a transition. The yellow blocks are referred to init process I , to the processes encoding the reactions, P_{r1} and P_{r2} , and to the context Cxt . As the figure puts in evidence, these two kinds of processes determine the structure of the link chain, from end to end, i.e. from the left τ to the right one. We could say that these processes form the *backbone* of the interaction. In contrast, the processes encoding the entities, P_{s1} , \bar{P}_{s2} , provide the solid links to be merged with the virtual links of the backbone (i.e. to be plugged in the backbone). Formally, we have

$$\langle \mathcal{A}, \gamma \rangle \xrightarrow{\tau \setminus r1 \setminus s1_i \setminus s1_o \setminus r2 \setminus s2_i \setminus s2_o \setminus cxt \setminus s1_i \setminus s1_o \setminus s2_i \setminus s2_o \setminus p1 \setminus s2_i \setminus s2_o \setminus p2 \setminus \tau} P'$$

Since the chain in the transition label is complete we can also write

$$\langle \mathcal{A}, \gamma \rangle \xrightarrow{\tau \setminus r1 \setminus s1_i \setminus s1_o \setminus r2 \setminus s2_i \setminus s2_o \setminus cxt \setminus s1_i \setminus s1_o \setminus s2_i \setminus s2_o \setminus p1 \setminus s2_i \setminus s2_o \setminus p2 \setminus \tau} P'$$

Example 15 outlines two different roles of the processes defining the translation of an interactive process: those processes encoding the reactions and the context provide the backbone of each transition, whereas the processes encoding the entities provide the resources needed for the communication to take place.

The flat function. For technical reasons, our transition labels are quite verbose; then, to simplify their processing, we introduce a function that takes a solid link chain and returns a simple string by eliminating all the channel matching pairs leaving just one placeholder for them. This transformation is harmless, in the sense that it retains all the information in the chain, because it is applied to complete chains only. The function $flat(\cdot)$ is defined inductively as follows:

$$flat(\epsilon) \triangleq \epsilon \quad flat(\alpha \setminus_{\beta}) \triangleq \begin{cases} \beta & \text{if } \beta \in reacts \cup \{cxt\} \cup prods \\ d & \text{if } \beta = d_i \text{ with } d \in decs \\ \epsilon & \text{otherwise} \end{cases}$$

$$flat(\alpha \setminus_{\beta} v) \triangleq flat(\alpha \setminus_{\beta}) :: flat(v)$$

where $::$ is the usual string concatenation operator.

For example, if we consider again the complete label

$$v = \tau \setminus_{r_1 \setminus_{s1_i \setminus_{s1_o} \setminus_{r_2 \setminus_{s2_i \setminus_{s2_o} \setminus_{cxt \setminus_{s1_i \setminus_{s1_o} \setminus_{s2_i \setminus_{s2_o} \setminus_{p_1 \setminus_{s2_i \setminus_{s2_o} \setminus_{p_2 \setminus_{\tau}}}}}}}}}}}} \tau$$

from Example 15, we have

$$flat(v) = r_1 \ s1 \ r2 \ \overline{s2} \ cxt \ \widehat{s1} \ \widehat{s2} \ p_1 \ \widetilde{s2} \ p_2.$$

It is then immediate to define the function $unflat$ to rebuild the complete label from the compact string (here we exploit again the half link and block notation):

$$unflat(x) \triangleq \begin{cases} x & \text{if } x \in reacts \cup \{cxt\} \cup prods \\ x_i \setminus_{x_o} & \text{if } x \in decs \end{cases}$$

$$unflat(x_1 \dots x_n) \triangleq \tau \setminus_{unflat(x_1)} \dots \setminus_{unflat(x_n)} \tau$$

It is immediate to check that for any complete label v of our processes we have $v = unflat(flat(v))$.

With the next proposition, we analyse the structure of a cCNA process encoding of a reactive process after one transition step. In the following four statements, for brevity, we let $\mathcal{A} = (S, A)$ be a RS, and let $\pi = (\gamma, \delta)$ be an extended interactive process in A , with $\gamma = \{C_i\}_{i \in \mathbb{N}}$ and $\delta = \{D_i\}_{i \in \mathbb{N}}$.

Proposition 16 (Correctness 1). *Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ with*

$$P = (\nu names) \left(I \mid \prod_{a \in A} P_a \mid Cxt^1 \mid \prod_{s \in C_0} P_s \mid \prod_{s \notin C_0} \overline{P}_s \right).$$

If there exists P' such that $P \xrightarrow{v} P'$, it holds that:

1. $v = \tau \setminus_{\tau} \dots \tau \setminus_{\tau}$, and
2. $P' = (\nu names) (I \mid \prod_{a \in A} P_a \mid Cxt^2 \mid \prod_{s \in C_1 \cup D_1} P_s \mid \prod_{s \notin C_1 \cup D_1} \overline{P}_s).$

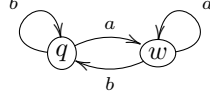


Figure 3: Minimal deterministic labelled transition system.

Moreover, given $\pi^1 = (\gamma^1, \delta^1)$, we have $P' = \llbracket \mathcal{A}, \gamma^1 \rrbracket$.

Now, we extend the previous result to a series of transitions.

Corollary 17 (Correctness 2). *Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ and $j \geq 1$. If there exists P'' such that $P \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau^j} P''$, then letting $\pi^j = (\gamma^j, \delta^j)$ we have $P'' = \llbracket \mathcal{A}, \gamma^j \rrbracket$.*

With the following propositions, we prove that, given a RS $\mathcal{A} = (S, A)$ and an extended interactive process $\pi = (\gamma, \delta)$, then the *cCNA* process $\llbracket \mathcal{A}, \gamma \rrbracket$ can simulate all the evolutions of π .

Proposition 18 (Completeness 1). *Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ and $\pi^1 = (\gamma^1, \delta^1)$. Then, $P \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau} P' = \llbracket \mathcal{A}, \gamma^1 \rrbracket$.*

Now, we extend the previous result to a series of transitions.

Corollary 19 (Completeness 2). *Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ and $\pi^j = (\gamma^j, \delta^j)$. Then, $P \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau^j} P'' = \llbracket \mathcal{A}, \gamma^j \rrbracket$.*

5. Examples

Hereafter, we shall often omit topmost restrictions but shall take into account only transitions whose labels are complete chains, i.e. they do not contain virtual links and start/end with the τ action symbol.

5.1. Labelled transition system

This example is inspired by the example in [1], where a deterministic transition system is coded in the Reaction System framework. Here we consider the minimal deterministic transition system in Figure 3. In our *cCNA* encoding, q , w , a , and b became the *entities*, the context can only provide a and b and the reaction rules are as follows:

$$\begin{array}{ll}
 1 & (\{q, a\}, \{w, b\}, \{q\}) \\
 3 & (\{w, a\}, \{q, b\}, \{q\})
 \end{array}
 \quad
 \begin{array}{ll}
 2 & (\{q, b\}, \{w, a\}, \{w\}) \\
 4 & (\{w, b\}, \{q, a\}, \{w\})
 \end{array}$$

Encoding of the rules. The encoding of the rules for reactions is given in a parametric way:

$$P_n(q, b, w, a, q) \triangleq \pi_n(q, b, w, a, q) \cdot P_n(q, b, w, a, q) + \sum_{x \in \{\bar{q}, \bar{b}, w, a\}} \pi'_n(x) \cdot P_n(q, b, w, a, q)$$

where

$$\begin{aligned} \pi_n(q, b, w, a, q) &\triangleq r_n \setminus_{q_i} \square \setminus_{q_o} \square \setminus_{b_i} \square \setminus_{b_o} \square \setminus_{\bar{w}_i} \square \setminus_{\bar{w}_o} \square \setminus_{\bar{a}_i} \square \setminus_{\bar{a}_o} \square \setminus_{r_{n+1}} \square \setminus_{p_n} \square \setminus_{\tilde{q}_i} \square \setminus_{\tilde{q}_o} \square \setminus_{p_{n+1}} \\ \pi'_n(x) &\triangleq r_n \setminus_{x_i} \square \setminus_{x_o} \square \setminus_{r_{n+1}} \square \setminus_{p_n} \square \setminus_{p_{n+1}} \end{aligned}$$

Then we have

$$\begin{aligned} P_1 &\triangleq P_1(q, a, w, b, q) & P_3 &\triangleq P_3(w, a, q, b, w) \\ P_2 &\triangleq P_2(q, b, w, a, w) & P_4 &\triangleq P_4(w, b, q, a, q) \end{aligned}$$

and we put, as usual, $r_5 = cxt$ and $p_5 = \tau$.

Encoding of the entities. As for reactions, also the encoding of the entities is given in a parametric way. Here we differentiate the encoding for the entities that are not provided by the context and that can be produced by the reactions, and the ones that can be provided by the context and that are not produced by the reactions.

Here, for the entities q and w that are not provided by the context, we let:

$$\begin{aligned} P_q &\triangleq E(q, \tilde{q}) & \bar{P}_q &\triangleq E(\bar{q}, \tilde{q}) \\ P_w &\triangleq E(w, \tilde{w}) & \bar{P}_w &\triangleq E(\bar{w}, \tilde{w}) \end{aligned}$$

where:

$$\begin{aligned} E(q, \tilde{q}) &\triangleq \sum_{h=1}^3 (q_i \setminus_{q_o} \square \setminus_{\square})^h \tilde{q}_i \setminus_{\tilde{q}_o} \square \cdot P_q \\ &+ \sum_{h=1}^3 (q_i \setminus_{q_o} \square \setminus_{\square})^h \bar{P}_q \end{aligned}$$

In fact the presence/absence of q and w will be exploited by at least one rule and at most three rules.

Here, for the entities a and b that can be provided by the context but not produced by the rules, we let:

$$\begin{aligned} P_a &\triangleq E(a, \hat{a}, \underline{a}) & \bar{P}_a &\triangleq E(\bar{a}, \hat{a}, \underline{a}) \\ P_b &\triangleq E(b, \hat{b}, \underline{b}) & \bar{P}_b &\triangleq E(\bar{b}, \hat{b}, \underline{b}) \end{aligned}$$

where:

$$\begin{aligned} E(a, \hat{a}, \underline{a}) &\triangleq \sum_{h=1}^3 (a_i \setminus_{a_o} \square \setminus_{\square})^h \hat{a}_i \setminus_{\hat{a}_o} \square \cdot P_a \\ &+ \sum_{h=1}^3 (a_i \setminus_{a_o} \square \setminus_{\square})^h \underline{a}_i \setminus_{\underline{a}_o} \square \cdot \bar{P}_a \end{aligned}$$

Finally, for the context, the encoding follows:

$$Cxt \triangleq cxt \setminus_{\hat{a}_i} \square \setminus_{\hat{a}_o} \square \setminus_{\hat{b}_i} \square \setminus_{\hat{b}_o} \square \setminus_{p_1} \square \cdot Cxt + cxt \setminus_{\bar{b}_i} \square \setminus_{\bar{b}_o} \square \setminus_{\bar{a}_i} \square \setminus_{\bar{a}_o} \square \setminus_{p_1} \square \cdot Cxt$$

Notice that we exploit here the capabilities of the process algebraic framework to define a nondeterministic, recursive context: We model the context to always offer either a or b , but never both the entities together. The reason is that in the other cases (providing both a and b or neither of them) would lead the system to be stuck because of the simplifications we have adopted in the other processes.

Now, we assume that we have an initial configuration containing the entities q and b :

$$Sys \triangleq I \mid P_q \mid \bar{P}_w \mid \bar{P}_a \mid P_b \mid P_1 \mid P_2 \mid P_3 \mid P_4 \mid Cxt.$$

Then, only the second reaction can be applied, and the transition carries the complete label v below

$$\tau \setminus r_1 \setminus \bar{a}_i \setminus \bar{a}_o \setminus r_2 \setminus q_i \setminus q_o \setminus b_i \setminus b_o \setminus \bar{w}_i \setminus \bar{w}_o \setminus \bar{a}_i \setminus \bar{a}_o \setminus r_3 \setminus \bar{a}_i \setminus \bar{a}_o \setminus r_4 \setminus \bar{w}_i \setminus \bar{w}_o \setminus cxt \setminus \bar{a}_i \setminus \bar{a}_o \setminus b_i \setminus b_o \setminus p_1 \setminus p_2 \setminus \bar{w}_i \setminus \bar{w}_o \setminus p_3 \setminus p_4 \setminus \tau$$

The parts in bold are provided by the entity processes, the other parts are provided by the processes encoding the reactions and by the process encoding the context (starting at cxt and ended at p_1 . In the label we can read that the rules 1 and 4 have been not executed because the entity a is absent, the rule 3 has been not applied because the entity w is absent, then only rule 2 has been applied, and it has produced the entity w . Also, the context provides entity a , that will be available in the next state, and not the entity b . Now, to let the label more readable, we show the result of the application of the function $flat(\cdot)$ to it:

$$r_1 \bar{a} r_2 q b \bar{w} \bar{a} r_3 \bar{a} r_4 \bar{w} cxt \hat{a} \underline{b} p_1 p_2 \tilde{w} p_3 p_4.$$

5.2. A biological toy example of gene expression

We consider a biological toy example in the style of gene's alternative splicing [17]. Alternative splicing is a regulated process during gene expression that results in a single gene coding for multiple proteins. In practice, particular exons of a gene may be included within or excluded from the final, processed messenger RNA (mRNA) produced from that gene.

where a gene a codes for a protein T when molecules G is present and C is absent, and in the opposite situation a codes for protein T' . This behavior is encoded in rules 1 and 2. Then, rule 3 codes for the production of C when proteins T and F are present, and T' absent; rule 4 codes for the production of G when proteins T' is present and F is absent.

Encoding of the rules. The encoding of the rules for reactions is given in a parametric way:

$$P_n(a, G, C, T) \triangleq \pi_n(a, G, C, T) \cdot P_n(a, G, C, T) + \sum_{x \in \{a, G, C\}} \pi'_n(x) \cdot P_n(a, G, C, T)$$

where

$$\begin{aligned} \pi_n(a, G, C, T) &\triangleq r_n \setminus \square_{a_i} \setminus \square_{a_o} \setminus \square_{G_i} \setminus \square_{G_o} \setminus \square_{C_i} \setminus \square_{C_o} \setminus \square_{r_{n+1}} \setminus \square_{p_n} \setminus \square_{\tilde{T}_i} \setminus \square_{\tilde{T}_o} \setminus \square_{p_{n+1}} \\ \pi'_n(x) &\triangleq r_n \setminus \square_{x_i} \setminus \square_{x_o} \setminus \square_{r_{n+1}} \setminus \square_{p_n} \setminus \square_{p_{n+1}} \end{aligned}$$

Now, to show a possible composition of a transition label, we assume a system where only the entities a , T' , and G are present:

$$Sys \triangleq I \mid P_a \mid \bar{P}_C \mid P_G \mid \bar{P}_F \mid \bar{P}_T \mid P_{T'} \mid P_1 \mid P_2 \mid P_3 \mid P_4 \mid Cxt$$

In the above configuration, reactions 1 and 4 can be applied, and also we assume that the context will provide the entity F , that will be available in the target configuration. Instead of showing the complete transition label, we give its flattened version obtained by applying the function $flat(\cdot)$:

$$r_1 \ a \ G \ \bar{C} \ r_2 \ G \ r_3 \ T' \ r_4 \ T' \ \bar{F} \ cxt \ \underline{C} \ \underline{G} \ \hat{F} \ p_1 \ \tilde{T} \ p_2 \ p_3 \ p_4 \ \tilde{G}.$$

The original label can then be reconstructed just applying the function $unflat(\cdot)$ to the string above.

In [16] we have shown a more complex example, by modeling a RS of a regulatory network for *lac* operon, presented in [4].

6. Bio-simulation

The classical notion of bisimulation for process algebras equates two processes when one process can simulate all the instructions executed by the other one and viceversa. In its weak formulation, internal instructions, i.e. non visible by external observers, are abstracted away. There are many variants of the bisimulation for process algebras, for example the barbed bisimulation [18] only consider the execution of invisible actions, and then equates two processes when they expose the same prefixes; for the mobile ambients [10], a process algebra equipped with a reduction semantics, a notion of behavioural equivalence equates two processes when they expose the same ambients [19].

There are some previous works based on bisimulation applied to models for biological systems. Barbuti et al [20] define a classical setting for bisimulation for two formalisms: the Calculus of Looping Sequences, which is a rewriting system, and the Brane Calculi, which is based on process calculi. Bisimulation is used to verify properties of the regulation of lactose degradation in *Escherichia coli* and the EGF signalling pathway. These calculi allow the authors to model membranes' behaviour. Cardelli et al [21] present two quantitative behavioral equivalences over species of a chemical reaction network with semantics based on ordinary differential equations. Bisimulation identifies a partition where each equivalence class represents the exact sum of the concentrations of the species belonging to that class. Bisimulation also relates species that have identical solutions at all time points when starting from the same initial conditions. Both the mentioned formalisms [20, 21] adopt a classical approach to bisimulation. Albeit the bisimulation is a powerful tool for verifying if the behaviour of two different software programs is indistinguishable, in the case of biological systems the classical bisimulation seems to be inappropriate, as it considers too many details. In fact, in a biological soup, a high number of interactions occur every seconds, and generally, biologists are only interested to analyse a small subset of them.

For this reason, we propose an alternative notion of bisimulation, that here-
 420 after we call *bio-simulation*, that allows us to compare two biological systems
 by restricting the observation to only the limited events of interest. This allows
 one to tailor the equivalence to different applications and purposes.

The transition labels of our systems record detailed information about all
 the reactions that have been applied in one transition, about the elements that
 425 acted as reagents, as inhibitors or that have been produced, or that have been
 provided by the context. All these information are stored in the label because
 they are necessary to compose a transition in a modular way. Depending on the
 application, only a suitable abstraction over the label can be of interest.

In a way, at each step of the bisimulation game, we want to query our
 430 transition labels to get only the information we care about. To this goal, we
 introduce a simple language that allows us to formulate detailed and partial
 queries about what happened in a single transition.

Example 20. For instance we would like to express properties about each step
 of the bio-simulation of a system like the ones below:

1. Has the entity s_i been used by rule r_j as reagent?
2. Has the entity s_i blocked the application of rule r_j ?
3. Has the entity s_i been produced by rule r_j ?
4. Has the entity s_i been produced by some rule?
5. Has the entity s_i been provided by the context?
- 440 6. Has the rule r_j not been applied?

As detailed before, in the following we assume that: (i) the context can
 be non-deterministic, otherwise it makes little sense to rely on bisimulation to
 observe the branching structure of system dynamics; (ii) we are interested in
 observing the names of the entities involved in the transitions and also the rules
 445 that have been applied, thus we assume top level restrictions are absent and
 rely on solid transitions only (with leftmost and rightmost silent actions).

6.1. Assertion language

Next, we introduce an assertion language that operates on simple strings and
 that combines regular expression operators with conjunction and disjunction.
 450 We let $syms = decs \cup reacts \cup prods \cup \{cxt\}$ be the set of symbolic names used
 in our assertion language.

Definition 21 (Assertion Language). Assertions are built from:

$$\begin{aligned} \zeta &::= \alpha \mid ? \mid [N] \\ F &::= \epsilon \mid \zeta \mid F :: F \mid F^+ \mid F^* \mid F \vee F \mid F \wedge F \end{aligned}$$

where $\alpha \in syms$ and $N \subseteq syms$.

Roughly, a *singleton* assertion ζ denotes either a string composed by a single symbol α (one of the symbols in the set *syms* for denoting a particular entity, rule, production or context), or the wildcard $?$ that stands for any symbol, or the pattern $[N]$ that stands for any of the strings composed by a single symbol in the set N . Clearly $?$ is just a shorthand for $[syms]$. We write $\mathbf{0}$ for $[\emptyset]$ and $[s_1, \dots, s_n]$ instead of $[\{s_1, \dots, s_n\}]$.

An *assertion* F is either the empty string ϵ , a singleton assertion ζ , the concatenation of two assertions $F_1 :: F_2$, the replication of F for 1 or more times F^+ , the replication of F for 0 or more times F^* , the disjunction of two assertions $F_1 \vee F_2$ or their conjunction $F_1 \wedge F_2$. We write \star as an abbreviation of $?$.

An assertion denotes a set of strings over the alphabet *syms* as expected.

Definition 22 (Semantics of Assertions). We define $\llbracket F \rrbracket \subseteq \wp(syms^*)$ by induction on the structure of F :

$$\begin{array}{llll} \llbracket \epsilon \rrbracket & \triangleq & \{\epsilon\} & \llbracket F_1 :: F_2 \rrbracket & \triangleq & \{\omega_1 :: \omega_2 \mid \omega_1 \in \llbracket F_1 \rrbracket \wedge \omega_2 \in \llbracket F_2 \rrbracket\} \\ \llbracket \alpha \rrbracket & \triangleq & \{\alpha\} & \llbracket F^+ \rrbracket & \triangleq & \llbracket F \rrbracket^+ \\ \llbracket ? \rrbracket & \triangleq & syms & \llbracket F^* \rrbracket & \triangleq & \llbracket F \rrbracket^* \\ \llbracket [N] \rrbracket & \triangleq & N & \llbracket F_1 \vee F_2 \rrbracket & \triangleq & \llbracket F_1 \rrbracket \cup \llbracket F_2 \rrbracket \\ & & & \llbracket F_1 \wedge F_2 \rrbracket & \triangleq & \llbracket F_1 \rrbracket \cap \llbracket F_2 \rrbracket \end{array}$$

Definition 23 (Satisfaction as Membership). Let v be a transition label, and F be an assertion. We write $v \models F$ (read as the transition label v satisfies the assertion F) if $flat(v) \in \llbracket F \rrbracket$, otherwise we write $v \not\models F$ (or also $v \models \neg F$) and say that F does not hold at v .

Given two transition labels v, w we write $v \equiv_F w$ if $v \models F \Leftrightarrow w \models F$, i.e. if both v, w satisfy F or they do not.

Example 24. The formulas corresponding to the sample queries listed in Example 20 are as follows:

1. $\star :: r_j :: [s_1, \dots, s_n]^* :: s_i :: [s_1, \dots, s_n]^* :: [\bar{s}_1, \dots, \bar{s}_n]^+ :: \star$
2. $\star :: r_j :: [s_i, \bar{s}_i] :: r_{j+1} :: \star$
3. $\star :: p_j :: [ents]^* :: \tilde{s}_i :: \star$
4. $\star :: \tilde{s}_i :: \star$
5. $\star :: \hat{s}_i :: \star$
6. $\star :: r_j :: ? :: r_{j+1} :: \star$

where in 1, 2, 6 we exploit the fact that in a reaction (R, I, P) the sets R of reactants and I of inhibitors are non empty, so that if there is only one symbol between the occurrence of r_j and r_{j+1} it means the reaction r_j has not been applied. Viceversa, if the reaction r_j has been applied the occurrence of r_j must be followed by at least one of the symbols in $\{s_1, \dots, s_n\}$ and then by at least one of the symbols in $\{\bar{s}_1, \dots, \bar{s}_n\}$.

6.2. Bio-similarity and bio-logical equivalence

The notion of bio-simulation builds on the above language of assertions to parameterize the induced equivalence on the property of interest. Please recall that we have defined the behaviour of the contest in a non deterministic way, thus at each step, different possible sets of entities can be provided to the system and different sets of reaction can be enabled/disabled. Bio-simulation can thus be used to compare the behaviour of different systems that share some of the reactions or entities or also to compare the behaviour of the same set of reaction rules when different contexts are provided.

Definition 25 (Bio-similarity \sim_F). Given an assertion F , a *bio-simulation* R_F that respects F is a binary relation over cCNA processes such that, whenever $P R_F Q$ then:

- for any v, P' such that $P \xRightarrow{v} P'$ then there exist w, Q' such that $Q \xRightarrow{w} Q'$ with $v \equiv_F w$ and $P' R_F Q'$.
- for any w, Q' such that $Q \xRightarrow{w} Q'$ then there exist v, P' such that $P \xRightarrow{v} P'$ with $v \equiv_F w$ and $P' R_F Q'$.

We let \sim_F denote the largest bio-simulation and we say that P is *bio-similar* to Q , with respect to F , if $P \sim_F Q$.

Remark 26. Please remember that the notation $P \xRightarrow{v} P'$ refers to ordinary transitions $P \xrightarrow{v} P'$ where v is a complete chain (solid and with τ actions at the extremes). The double arrow notation should not be confused with the notation for weak transitions commonly found in the literature on process algebras.

Remark 27. An alternative way to look at a bio-simulation that respects F is to define it as an ordinary bisimulation over the transition system labelled over $\{F, \neg F\}$ obtained by transforming each transition $P \xRightarrow{v} P'$ such that $v \models F$ into $P \xrightarrow{F} P'$ and each transition $P \xRightarrow{v} P'$ such that $v \not\models F$ into $P \xrightarrow{\neg F} P'$.

It can be easily shown that the identity relation is a bio-simulation and that bio-simulations are closed under (relational) inverse, composition and union and that, as a consequence, bio-similarity is an equivalence relation.

Now, we introduce a slightly modified version of the Hennessy Milner Logic [13], called bioHML; due to the reasons we explained above, we do not want to look at the complete transition labels, thus we rely on our simple assertion language to make it parametric to the assertion F of interest:

Definition 28 (BioHML). Let F be an assertion, then the set of bioHML formulas G that respects F are built by the following syntax:

$$\begin{aligned} \chi &::= F \mid \neg F \\ G, H &::= \mathbf{t} \mid \mathbf{f} \mid G \wedge G \mid G \vee G \mid \langle \chi \rangle G \mid [\chi] G \end{aligned}$$

Remark 29. An alternative way to look at bioHML formulas is as ordinary HML formulas over the set of labels $\{F, \neg F\}$.

As usual, the semantics of a bioHML formula is the set of processes that satisfy it.

Definition 30 (Semantics of BioHML). We define $\llbracket G \rrbracket \subseteq \mathcal{P}$ by induction on the structure of G :

$$\begin{aligned} \llbracket \mathbf{t} \rrbracket &\triangleq \mathcal{P} & \llbracket G \wedge H \rrbracket &\triangleq \llbracket G \rrbracket \cap \llbracket H \rrbracket \\ \llbracket \mathbf{f} \rrbracket &\triangleq \emptyset & \llbracket G \vee H \rrbracket &\triangleq \llbracket G \rrbracket \cup \llbracket H \rrbracket \end{aligned}$$

$$\begin{aligned} \llbracket \langle \chi \rangle G \rrbracket &\triangleq \{P \in \mathcal{P} : \exists v, P'. P \xrightarrow{v} P' \text{ with } v \models \chi \text{ and } P' \in \llbracket G \rrbracket\} \\ \llbracket [\chi] G \rrbracket &\triangleq \{P \in \mathcal{P} : \forall v, P'. P \xrightarrow{v} P' \text{ implies } v \models \chi \text{ and } P' \in \llbracket G \rrbracket\} \end{aligned}$$

We write $P \models G$ (P satisfies G) if and only if $P \in \llbracket G \rrbracket$.

Negation is not included in the syntax, but the converse \overline{G} of a bioHML formula G can be easily defined inductively in the same way as for HML logic.

Definition 31 (Converse). Given a bioHML formula G we define its converse \overline{G} as follows:

$$\begin{aligned} \overline{\mathbf{t}} &\triangleq \mathbf{f} & \overline{G \wedge H} &\triangleq \overline{G} \vee \overline{H} & \overline{\langle \chi \rangle G} &\triangleq [\chi] \overline{G} \\ \overline{\mathbf{f}} &\triangleq \mathbf{t} & \overline{G \vee H} &\triangleq \overline{G} \wedge \overline{H} & \overline{[\chi] G} &\triangleq \langle \chi \rangle \overline{G} \end{aligned}$$

We observe that, as expected, for any bioHML formula G and process P we have $\overline{\overline{G}} = G$ and $P \models \overline{G}$ iff $P \not\models G$.

We let \mathcal{L}_F be the set of all bioHML formulas that respects F and we say that two processes P, Q are bio-logically equivalent w.r.t. F , written $P \equiv_{\mathcal{L}_F} Q$, when P and Q satisfy the exactly the same bioHML formulas in \mathcal{L}_F , i.e. when for any $G \in \mathcal{L}_F$ we have $P \models G \Leftrightarrow Q \models G$.

Finally, we extend the classical result establishing the correspondence between the logical equivalence induced by HML with bisimilarity for proving that bio-similarity coincides with bio-logical equivalence.

Theorem 32 (Correspondence). $\sim_F = \equiv_{\mathcal{L}_F}$

6.3. Bio-simulation at work

We will show how bio-simulation works. For the sake of space, we consider a very simple example with only two reactions. Reaction P_1 requires G to produce C ; reaction P_2 requires C to produce G ; both reactions have H as inhibitor. Now, we set two systems defined by the same two reactions, with the two different initial configuration, and with two different context definitions. The two reactions work as follows (where we omit to specify the cases where H is present, as they will never happen):

$$\begin{aligned} P_1 &\triangleq \tau \backslash_{G_i}^{\square} \backslash_{\overline{H}_i}^{\square} \backslash_{\overline{H}_o}^{\square} \backslash_{r_2}^{\square} \backslash_{\tilde{C}_i}^{\square} \backslash_{p_2}^{\square} \cdot P_1 + \tau \backslash_{\overline{G}_i}^{\square} \backslash_{\overline{G}_o}^{\square} \backslash_{r_2}^{\square} \backslash_{p_2}^{\square} \cdot P_1 \\ P_2 &\triangleq r_2 \backslash_{C_i}^{\square} \backslash_{\overline{C}_o}^{\square} \backslash_{\overline{H}_i}^{\square} \backslash_{\overline{H}_o}^{\square} \backslash_{cxt}^{\square} \backslash_{\tilde{G}_i}^{\square} \backslash_{\tau}^{\square} \cdot P_2 + r_2 \backslash_{\overline{C}_i}^{\square} \backslash_{\overline{C}_o}^{\square} \backslash_{cxt}^{\square} \backslash_{p_2}^{\square} \backslash_{\tau}^{\square} \cdot P_2 \end{aligned}$$

The two contexts follow :

$$Ctx_1 \triangleq \text{cxt} \backslash_{\widehat{C}_i} \square \backslash_{\widehat{C}_o} \square \backslash_{\widehat{G}_i} \square \backslash_{\widehat{H}_i} \square \backslash_{p_1} \cdot Ctx_1 + \text{cxt} \backslash_{\underline{C}_i} \square \backslash_{\underline{C}_o} \square \backslash_{\underline{G}_i} \square \backslash_{\underline{H}_i} \square \backslash_{p_1} \cdot Ctx_1$$

$$Ctx_2 \triangleq \text{cxt} \backslash_{\widehat{G}_i} \square \backslash_{\widehat{C}_o} \square \backslash_{\widehat{C}_i} \square \backslash_{\widehat{H}_i} \square \backslash_{p_1} \cdot Ctx_2 + \text{cxt} \backslash_{\underline{G}_i} \square \backslash_{\underline{C}_o} \square \backslash_{\underline{C}_i} \square \backslash_{\underline{H}_i} \square \backslash_{p_1} \cdot Ctx_2$$

The definition of the processes encoding G and C is similar, and it is given in a parametric way:

$$P(G) \triangleq G_i \backslash_{G_o} \cdot \overline{P}(G) \quad \overline{P}(G) \triangleq \overline{G}_i \backslash_{\overline{G}_o} \square \backslash_{\overline{G}_i} \square \backslash_{\overline{G}_o} P(G)$$

and we have $P_G \triangleq P(G)$, $P_C \triangleq P(C)$, then $\overline{P}_H \triangleq \overline{H}_i \backslash_{\overline{H}_o} \cdot \overline{P}_H + \overline{H}_i \backslash_{\overline{H}_o} \square \backslash_{\overline{H}_i} \square \backslash_{\overline{H}_i} \cdot \overline{P}_H$, as H is neither produced nor provided by the context. Then, the initial configuration of system Sys_1 includes C and not G and the context can only provide G , the initial configuration of system Sys_2 includes G and not C and the context can only provide C :

$$\begin{aligned} Sys_1 &\triangleq I \mid P_1 \mid P_2 \mid P_G \mid \overline{P}_C \mid \overline{P}_H \mid Ctx_1 \\ Sys_2 &\triangleq I \mid P_1 \mid P_2 \mid \overline{P}_G \mid P_C \mid \overline{P}_H \mid Ctx_2 \end{aligned}$$

Fig. 4 shows the labelled transition system of Sys_1 , with initial state only containing G , and the transition system of Sys_2 , with initial state only containing C , limited to the complete and solid labels. As before we show the output of the $flat(\cdot)$ function applied to the transition labels:

$$\begin{aligned} \ell_1 &\triangleq r_1 \ G \ \overline{H} \ r_2 \ \overline{C} \ \text{cxt} \ \widehat{C} \ \underline{G} \ \underline{H} \ p_1 \ \widetilde{C} \ p_2 \quad \ell'_1 \triangleq r_1 \ G \ \overline{H} \ r_2 \ \overline{C} \ \text{cxt} \ \underline{C} \ \underline{G} \ \underline{H} \ p_1 \ \widetilde{C} \ p_2 \\ \ell_2 &\triangleq r_1 \ \overline{G} \ r_2 \ C \ \overline{H} \ \text{cxt} \ \widehat{C} \ \underline{G} \ \underline{H} \ p_1 \ p_2 \ \widetilde{G} \quad \ell'_2 \triangleq r_1 \ \overline{G} \ r_2 \ C \ \overline{H} \ \text{cxt} \ \underline{C} \ \underline{G} \ \underline{H} \ p_1 \ p_2 \ \widetilde{G} \\ \ell_3 &\triangleq r_1 \ G \ \overline{H} \ r_2 \ C \ \overline{H} \ \text{cxt} \ \widehat{C} \ \underline{G} \ \underline{H} \ p_1 \ \widetilde{C} \ p_2 \ \widetilde{G} \\ \ell'_3 &\triangleq r_1 \ G \ \overline{H} \ r_2 \ C \ \overline{H} \ \text{cxt} \ \underline{C} \ \underline{G} \ \underline{H} \ p_1 \ \widetilde{C} \ p_2 \ \widetilde{G} \end{aligned}$$

The labels $b_i \ b'_i$, with $i \in \{1, 2, 3\}$ can be obtained by labels ℓ_i, ℓ'_i by substituting G with C and viceversa.

Now, it easy to check that Sys_1 and Sys_2 are bio-similar w.r.t the property F saying that G and C are simultaneously produced, formally: $Sys_1 \sim_F Sys_2$ with $F = \star :: \widetilde{G} :: \star \wedge \star :: \widetilde{C} :: \star$.

On the contrary, $Sys_1 \not\sim_{F'} Sys_2$ with $F' = \star :: \widetilde{C} :: \star$, because it happens that both transition labels ℓ_1 and ℓ'_1 , in Sys_1 , record the production of C , whereas transition labels b_1 and b'_1 do not. In fact, the bioHML formula $G \triangleq \langle F' \rangle t$ can be used to distinguish Sys_1 from Sys_2 , as $Sys_1 \models G$ and $Sys_2 \not\models G$.

7. Enhanced Reaction Systems

Our encoding increases the expressivity of RS concerning: the possibility of alternative behaviour of mutated entities, and the communication between two different Reaction Systems. It is important to note that, when the context is deterministic, our encoding guarantees that from each state, in the cCNA transition system, only one state is reachable, as the dynamics is totally deterministic.

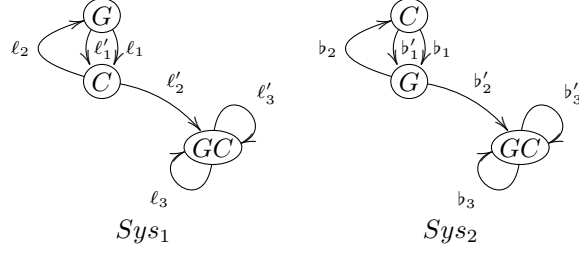


Figure 4: The two labelled transition systems of Sys_1 and Sys_2 .

$rs1$	$rs2$
$a_1 = (s, , x)$	$a_2 = (y, , s)$

Table 1: The two Reaction Systems $rs1$ and $rs2$.

7.1. Mutating entities

In RS, when an entity is present, it can potentially be involved in each reactions where it is required. With a few more lines of code, in cCNA it is possible to describe the behaviour of a mutation of an entity, in a way that the mutated version of the entity can take part to only a subset of the rules requiring the *normal version* of the entity. For example, let us assume that entity $s1$ is consumed by reactions $a1$ and $a2$. Reaction $a1$ produces also $s1$ if $s2$ is present, otherwise $a1$ produces a mutated version of $s1$, say $s1'$. When $s1'$ is produced, reaction $a2$ behaves in the same way as if $s1$ would be absent, whereas $a2$ recognises the presence of $s1'$ and behaves in the same way as if $s1$ would be present. Technically, in both cases it is enough to add one more nondeterministic choice in the code of P_{a1} and P_{a2} .

7.2. Communicating Reaction Systems

We sketch how it is possible to program two RSs encodings, in a way that the entities that usually come from the context of one RS will be provided instead from the other RS.

Example 33. Let $rs1$ and $rs2$ be two RSs, defined by the rules in Table 1. Now, we set our example such that the two contexts, for $rs1$ and $rs2$, do not provide any entities. We also assume that entity s in $rs1$ is provided by $rs2$, as $rs2$ produces a quantity of s that is enough for $rs1$ and $rs2$. For technical reasons, we can not use the same name for s in both the two RSs, then we use the name ss in $rs2$. We need to modify our translation technique to suite this new setting. As we do not model contexts, we introduce *dummy* channel names dx and dss to model the absence of entities. Also, thanks to the simplicity of the example, we can leave out the use of the p_i channels. This streamlining does not affect the programming technique we propose to make two RSs communicate. First, we translate the reaction in $rs1$:

$$[[a_1]] \triangleq P_{a_1} \triangleq \tau \backslash_{s_i} \square \backslash_{s_o} \square \backslash_{\tilde{x}_o} \square \backslash_{a_2} \cdot P_{a_1} + \tau \backslash_{\tilde{s}_i} \square \backslash_{\tilde{s}_o} \square \backslash_{dx_i} \square \backslash_{a_2} \cdot P_{a_1}$$

Please note, that prefixes of process P_{a_1} end with the channel name a_2 , as the link chain is now connected with the reaction of $rs2$. The translation for the entities follows.

$$\begin{aligned} \llbracket s \rrbracket &\triangleq \frac{P_s}{\overline{P_s}} \triangleq \frac{s_i \backslash \square \backslash \widehat{s_i} \backslash \widehat{s_o} \cdot P_s + s_i \backslash s_o \cdot \overline{P_s}}{\overline{P_s}} \triangleq \frac{\overline{s_i} \backslash \square \backslash \widehat{s_i} \backslash \widehat{s_o} \cdot P_s + \overline{s_i} \backslash s_o \cdot \overline{P_s}}{\overline{P_s}} \\ \llbracket x \rrbracket &\triangleq \frac{P_x}{\overline{P_x}} \triangleq \frac{\widetilde{x_i} \backslash \widetilde{x_o} \cdot P_x + dx_i \backslash dx_o \cdot \overline{P_x}}{\overline{P_x}} \triangleq \frac{\widetilde{x_i} \backslash \widetilde{x_o} \cdot P_x + dx_i \backslash dx_o \cdot \overline{P_x}}{\overline{P_x}} \end{aligned}$$

The translation for $rs2$ follows.

$$\llbracket a_2 \rrbracket \triangleq P_{a_2} \triangleq a_2 \backslash \square \backslash y_o \backslash \square \backslash \widetilde{ss_i} \backslash \widetilde{ss_o} \backslash \tau \cdot P_{a_2} + a_2 \backslash \square \backslash \overline{y_o} \backslash \square \backslash dss_o \backslash \tau \cdot P_{a_2}$$

In the translation of the entities in $rs2$, we introduce the mechanism that allows the entity s (ss in $rs2$) to be provided in $rs1$. Every time ss is produced in $rs2$, a virtual link is created to synchronise with $rs1$ on link $\widehat{s_i} \backslash \widehat{s_o}$:

$$\begin{aligned} \llbracket ss \rrbracket &\triangleq \frac{P_{ss}}{\overline{P_{ss}}} \triangleq \frac{\widetilde{ss_i} \backslash \square \backslash \widehat{s_i} \backslash \widehat{s_o} \cdot P_{ss} + dss_i \backslash dss_o \cdot \overline{P_{ss}}}{\overline{P_{ss}}} \triangleq \frac{\widetilde{ss_i} \backslash \square \backslash \widehat{s_i} \backslash \widehat{s_o} \cdot P_{ss} + dss_i \backslash dss_o \cdot \overline{P_{ss}}}{\overline{P_{ss}}} \\ \llbracket y \rrbracket &\triangleq \frac{P_y}{\overline{P_y}} \triangleq \frac{y_i \backslash y_o \cdot \overline{P_y}}{\overline{P_y}} \triangleq \frac{y_i \backslash y_o \cdot \overline{P_y}}{\overline{P_y}} \end{aligned}$$

We now assume that the initial system is $S \triangleq (\nu names)(P_{a_1} | P_{a_2} | P_s | P_y | \overline{P_x} | \overline{P_{ss}})$, i.e. only entities s and y are present. Now, the only possible transition has the following label (that we report without restriction):

$$\tau \backslash s_i \backslash s_o \backslash \widetilde{x_i} \backslash \widetilde{x_o} \backslash a_2 \backslash y_i \backslash y_o \backslash \widetilde{ss_i} \backslash \widehat{s_i} \backslash \widehat{s_o} \backslash \widetilde{ss_o} \backslash \tau,$$

where the black links belong to the prefixes of P_{a_1} , and P_{a_2} , the blue links belong to P_s , the gray links belong to P_y , and $\overline{P_x}$ and the red links belong to $\overline{P_{ss}}$. After the execution, the entity s is still present in $rs1$ as it has been provided by $rs2$.

As we have briefly sketched, our model of two *communicating Reaction Systems* can enable the study of the behaviour of one RS in relation to another one. Thus, the products of the reactions of one RS can become the input for another one. This could allow for a modular approach to modeling complex systems, by composing different Reaction Systems.

8. Conclusion

In this paper we have introduced cCNA, a variant of the `link-calculus` where prefixes are link chains and no more single links, as it was briefly described in the future work section in [9]. This variant allowed us to define a faithful embedding of Reaction Systems, an emerging formalism to model computationally biochemical systems. This translation shows several benefits. For instance, in our

paper the context of a RS can work non deterministically and it is recursively defined. Also, entity mutations can be expressed easily and different reaction systems can communicate between them.

We have then modified the classical notion (in concurrency) of bisimulation making it more suitable for proving properties of Reaction Systems in our framework. We have defined a new assertion language, which allows us to specify the properties to be verified, and have shown that it extends Hennessy-Milner logic to our framework. We believe that our work can make possible to investigate how to integrate our methodology with other formal techniques to prove properties of the modeled systems [22, 23, 24].

We are confident that our embedding can contribute to extend the applications of Reaction Systems to diverse fields of computer science, and life sciences.

As future work, we plan to implement a prototype of our framework, with an automatic translation from RSs to `link`-calculus. We can also exploit the implementation of the symbolic semantics of the `link`-calculus [25] that can be found in [26].

We also plan to build on the work in this paper to extend the framework of RSs towards a model where different RSs are allowed to communicate.

References

- [1] R. Brijder, A. Ehrenfeucht, M. Main, G. Rozenberg, A tour of reaction systems, *International Journal of Foundations of Computer Science* 22 (07) (2011) 1499–1517. [1](#), [2](#), [5.1](#)
- [2] A. Męski, W. Penczek, G. Rozenberg, Model checking temporal properties of reaction systems, *Information Sciences* 313 (2015) 22–42. [doi:10.1016/j.ins.2015.03.048](#). [1](#), [2](#)
- [3] S. Azimi, B. Iancu, I. Petre, Reaction system models for the heat shock response, *Fundamenta Informaticae* 131 (3-4) (2014) 299–312. [doi:10.3233/FI-2014-1016](#). [1](#), [2](#)
- [4] L. Corolli, C. Maj, F. Marinia, D. Besozzi, G. Mauri, An excursion in reaction systems: From computer science to biology, *Theoretical Computer Science* 454 (2012) 95–108. [1](#), [2](#), [5.2](#)
- [5] S. Azimi, Steady states of constrained reaction systems, *Theor. Comput. Sci.* 701 (C) (2017) 20–26. [doi:10.1016/j.tcs.2017.03.047](#). [1](#), [2](#)
- [6] R. Barbuti, R. Gori, F. Levi, P. Milazzo, Investigating dynamic causalities in reaction systems, *Theor. Comput. Sci.* 623 (2016) 114–145. [1](#), [2](#)
- [7] F. Okubo, T. Yokomori, [The computational capability of chemical reaction automata](#), *Natural Computing* 15 (2) (2016) 215–224. [doi:10.1007/s11047-015-9504-7](#). URL <https://doi.org/10.1007/s11047-015-9504-7> [1](#), [2](#)

- [8] C. Bodei, L. Brodo, R. Bruni, Open multiparty interaction, in: Recent Trends in Algebraic Development Techniques, 21st International Workshop, WADT 2012, Vol. 7841 of Lecture Notes in Computer Science, Springer, 2012, pp. 1–23. [1](#), [3](#)
- [9] C. Bodei, L. Brodo, R. Bruni, A formal approach to open multiparty interactions, Theoretical Computer Science 763 (2019) 38–65. [1](#), [3](#), [3](#), [8](#)
- [10] L. Cardelli, A. D. Gordon, Mobile ambients, Theoretical Computer Science 240 (1) (2000) 177–213. [1](#), [6](#)
- [11] L. Brodo, On the expressiveness of pi-calculus for encoding mobile ambients, Mathematical Structures in Computer Science 28 (2) (2018) 202–240. [1](#)
- [12] C. Bodei, L. Brodo, R. Bruni, D. Chiarugi, A flat process calculus for nested membrane interactions, Sci. Ann. Comp. Sci. 24 (1) (2014) 91–136. [1](#)
- [13] M. Hennessy, R. Milner, On observing nondeterminism and concurrency, in: J. de Bakker, J. van Leeuwen (Eds.), Automata, Languages and Programming, Vol. 85 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 1980, pp. 299–309. [1](#), [6.2](#)
- [14] A. Bernini, L. Brodo, P. Degano, M. Falaschi, D. Hermith, Process calculi for biological processes, Natural Computing 17 (2) (2018) 345–373. [1](#)
- [15] A. Bogdan, C. Gabriel, Controlled reversibility in reaction systems, in: M. Gheorghe, G. Rozenberg, A. Salomaa, Z. Claudio (Eds.), Membrane Computing, Springer International Publishing, 2018, pp. 40–53. [1](#)
- [16] L. Brodo, R. Bruni, M. Falaschi, Embedding reaction systems into link-calculus, in: M. Alvim, K. Chatzikokolakis, C. Olarte, F. Valencia (Eds.), The Art of Modelling Computational Systems: A Journey from Logic and Concurrency to Security and Privacy, Vol. 11760 of Lecture Notes in Computer Science, Springer Berlin, 2019, pp. 68–85. [1](#), [5.2](#)
- [17] J. D. Watson, T. A. Baker, S. P. Bell, Molecular Biology of the Gene, Pearson Education, USA, 2013. [5.2](#)
- [18] R. Milner, D. Sangiorgi, Barbed bisimulation, in: W. Kuich (Ed.), Automata, Languages and Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, 1992, pp. 685–695. [6](#)
- [19] A. Gordon, L. Cardelli, Equational properties of mobile ambients, Mathematical Structures in Computer Science 13 (3) (2003) 371–408. [6](#)
- [20] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, A. Troina, Bisimulations in calculi modelling membranes, Formal Aspects of Computing 20 (4) (2008) 351–377. [6](#)

- [21] L. Cardelli, M. Tribastone, M. Tschaikowski, A. Vandin, Forward and backward bisimulations for chemical reaction networks, in: 26th International Conference on Concurrency Theory, CONCUR 2015, Vol. 42, Schloss Dagstuhl- Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, 2015, pp. 226–239. doi:10.4230/LIPIcs.CONCUR.2015.226. 6
- [22] D. Chiarugi, M. Falaschi, D. Hermith, C. Olarte, L. Torella, Modelling non-markovian dynamics in biochemical reactions, BMC Systems Biology 9 (S-3) (2015) S8. 8
- [23] C. Olarte, D. Chiarugi, M. Falaschi, D. Hermith, A proof theoretic view of spatial and temporal dependencies in biochemical systems, Theor. Comput. Sci. 641 (2016) 25–42. 8
- [24] C. Bodei, L. Brodo, R. Gori, F. Levi, A. Bernini, D. Hermith, A static analysis for Brane Calculi providing global occurrence counting information, Theoretical Computer Science 696 (2017) 11–51. 8
- [25] L. Brodo, C. Olarte, Symbolic semantics for multiparty interactions in the link-calculus, in: Proc. of SOFSEM'17, Vol. 10139 of Lecture Notes in Computer Science, Springer, 2017, pp. 62–75. 8
- [26] C. Olarte, SiLVer: Symbolic links verifier, <http://subsell.logic.at/links/links-web/index.html> (Dec. 2018). 8

Appendix A. Omitted Proofs

In this section we report the proofs for the results in Section 4 and in Section 6.

Lemma 12. *Let $\mathcal{A} = (S, A)$ be a RS and let $\pi = (\gamma, \delta)$ be an extended interactive process in \mathcal{A} . Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ its cCNA translation. If exists P' such that $P \xrightarrow{(\nu \text{ names})v} P'$ is a transition of P , then*

1. *for each reaction $aj \in A$, the corresponding channels r_j and p_j appear in v ; for each entity $s \in S$, the corresponding channel s (suitably decorated) appear in v ; the channel cxt appears in v ;*
2. *for each reaction $aj \in A$ and each entity $s \in S$, each virtual link offered by processes P_a and Cxt is overlapped by exactly one solid link offered by processes representing entities.*

PROOF. We prove the two items separately:

1. by Definition 11, all the names that appear in the prefixes of any subprocess are in the set *names* and thus restricted. They include all the reaction names r_j , all the production names p_j , all the entity names s_i, s_o , in all their decorated versions, and the name cxt . Therefore the chain v must start and end with a τ action and cannot contain virtual links. The only prefix that starts with τ is the one of the recursive init process I (prefix $\tau \setminus_{r_1}$) and the only prefixes that end with τ are those associated to reaction a_u (as we have assumed that $p_{u+1} = \tau$, where u is the number of reactions). Then each prefix that starts with r_j involves p_j and r_{j+1} . Thus all the prefixes associated with reactions must be concatenated and also the prefix associated with the context (remember that $r_{u+1} = cxt$), forming the backbone of the label. Since the context processes is involved, then all entities processes are also involved. Then, all the processes I, P_a, P_s (or $\overline{P_s}$), and Cxt must participate to each transition.
2. for each reaction $aj \in A$, the cCNA code of P_{aj} leaves one virtual link between two solid links of the types $r_j \setminus \dots \setminus_{s_i} \setminus_{\square}^{s_o} \setminus \dots \setminus_{r_{j+1}} \dots p_j \setminus \dots \setminus_{p_{j+1}}$. Then, it derives that the process P_s , encoding the behaviour of entity s , can participate by filling the virtual link in the above transition by only offering one solid link of the form $s_i \setminus_{s_o}$. In fact, there is no other way to generate a solid chain from s_i to s_o . The same reasoning holds for the processes Cxt and for all the decorated versions of s_i, s_o .

Proposition 16 (Correctness 1). *Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ with*

$$P = (\nu \text{ names}) \left(I \mid \prod_{a \in A} P_a \mid Cxt^1 \mid \prod_{s \in C_0} P_s \mid \prod_{s \notin C_0} \overline{P_s} \right).$$

If there exists P' such that $P \xrightarrow{v} P'$, it holds that:

- 1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
- 720 1. $v = \tau \backslash \tau \dots \tau \backslash \tau$, and
2. $P' = (\nu \text{ names}) (I \mid \prod_{a \in A} P_a \mid Cxt^2 \mid \prod_{s \in C_1 \cup D_1} P_s \mid \prod_{s \notin C_1 \cup D_1} \bar{P}_s)$.

Moreover, given $\pi^1 = (\gamma^1, \delta^1)$, we have $P' = \llbracket \mathcal{A}, \gamma^1 \rrbracket$.

PROOF. First, we note that all the channels in the system are restricted, see Def. 11, then it holds that the transition labels are of the form $v = \tau \backslash \tau \dots \tau \backslash \tau$.
725 Now, by Definition 11 and by Lemma 12.1, all the channels r_j, p_j , with $j \in [1, \dots, u]$, and cxt_h , with $h \in [1, \dots, w]$, and all the annotated versions of s_i, s_o are restricted. Also, processes Cxt always requires the interaction with P_s on either on channels $\widehat{s}_i, \widehat{s}_o$ or on channels $\underline{s}_i, \underline{s}_o$. It derives that all the processes: P_a (coding the behaviour of reaction $a \in A$), P_s (coding the behaviour of entity $s \in S$), and Cxt (coding the behaviour of the context regarding all the entities)
730 have been involved in the transition.

For any process P_{aj} encoding a reaction aj we have the following cases:

- (a) if aj is applicable and it produces the entity s , the process P_{aj} provides a code of this type:

$$P_{aj} \triangleq r_j \backslash \dots \backslash \square_{r_{j+1}} \backslash \dots \backslash \square_{s_i} \backslash \dots \backslash \square_{s_o} \backslash \dots \backslash p_{j+1} \cdot P_{aj};$$

- (b) if aj is applicable and it consumes the entity s , the process P_{aj} provides a code of this type:

$$P_{aj} \triangleq r_j \backslash \dots \backslash \square_{s_i} \backslash \dots \backslash \square_{s_o} \backslash \dots \backslash \square_{r_{j+1}} \backslash \dots \backslash p_{j+1} \cdot P_{aj};$$

- (c) if aj is applicable and it requires the absence of the entity s , the process P_{aj} provides a code of this type:

$$P_{aj} \triangleq r_j \backslash \dots \backslash \square_{s_i} \backslash \dots \backslash \square_{s_o} \backslash \dots \backslash \square_{r_{j+1}} \backslash \dots \backslash p_{j+1} \cdot P_{aj};$$

- (d) if aj is not applicable, the process P_{aj} executes a code capturing either the absence of one of its reactants (case 1), or the presence of one of its inhibitors (case 2):
735

1. $P_{aj} \triangleq r_j \backslash \dots \backslash \square_{s_i} \backslash \dots \backslash \square_{s_o} \backslash \dots \backslash \square_{r_{j+1}} \backslash \dots \backslash p_{j+1} \cdot P_{aj};$
2. $P_{aj} \triangleq r_j \backslash \dots \backslash \square_{s_i} \backslash \dots \backslash \square_{s_o} \backslash \dots \backslash \square_{r_{j+1}} \backslash \dots \backslash p_{j+1} \cdot P_{aj}.$

Now, we consider the structure of the process Cxt^1 . By Definition 11, Cxt^1 is the unique process encoding the behaviour of the context regulating the supply
740 of any entity.

- (e) The code of the process Cxt^1 that provides s and not e has the following structure:

$$Cxt^1 \triangleq cxt \backslash \dots \backslash \square_{s_i} \backslash \dots \backslash \square_{s_o} \backslash \dots \backslash \square_{e_i} \backslash \dots \backslash p_1 \cdot Cxt^2.$$

The code executed by P_s has the following structure:

- (f) $P_s \triangleq \sum_{h,k \geq 0} (s_i \setminus \square \setminus \square)^h \widehat{s}_i \setminus \square \setminus \square (\widetilde{s}_i \setminus \square \setminus \square)^k . P_s$, if $s \in C_{i+1}$;
 (g) $P_s \triangleq \sum_{h \geq 0, k \geq 1} (s_i \setminus \square \setminus \square)^h \underline{s}_i \setminus \square \setminus \square (\widetilde{s}_i \setminus \square \setminus \square)^k . P_s$, if $s \notin C_{i+1}$
 (h) $P_s \triangleq \sum_{h \geq 0} (s_i \setminus \square \setminus \square)^h \underline{s}_i \setminus \square . \overline{P}_s$, if $s \notin C_{i+1}$

745 where, by Lemma 12.2, h is the number of reactions requiring the presence of s plus possibly some reactions not requiring s ; and k is the number of reactions producing s .

Similarly, the code executed by \overline{P}_s has the following structure:

- (f') $\overline{P}_s \triangleq \sum_{h,k \geq 0} (\widetilde{s}_i \setminus \square \setminus \square)^h \widehat{s}_i \setminus \square \setminus \square (\widetilde{s}_i \setminus \square \setminus \square)^k . P_s$, if $s \in C_{i+1}$;
 750 (g') $\overline{P}_s \triangleq \sum_{h \geq 0, k \geq 1} (\widetilde{s}_i \setminus \square \setminus \square)^h \underline{s}_i \setminus \square \setminus \square (\widetilde{s}_i \setminus \square \setminus \square)^k . P_s$, if $s \notin C_{i+1}$
 (h') $\overline{P}_s \triangleq \sum_{h \geq 0} (\widetilde{s}_i \setminus \square \setminus \square)^h \underline{s}_i \setminus \square . \overline{P}_s$, if $s \notin C_{i+1}$

where, by Lemma 12.2, h is the number of reactions requiring the absence of s plus possibly some reactions requiring s ; and k is the number of reactions producing s .

755 It is worth nothing that, depending on the presence (P_s) or the absence (\overline{P}_s) of each entity s , for each process P_a (encoding a reaction a) the choice between the execution of the reaction code (points (a), (b), (c)) or the code expressing that reaction a is not applicable (point (d)) is deterministic. Also, the building of the code of process Cxt (points (e), (f)), is univocally determined
 760 by the evolution of γ . It derives that the trend followed by the processes P_s (or \overline{P}_s) is also deterministic (points (f), (g), (h) or (f'), (g'), (h')), leading to $P' = \llbracket \mathcal{A}, \gamma^1 \rrbracket$.

Corollary 17 (Correctness 2). *Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ and $j \geq 1$. If there exists P'' such that $P \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau}^j P''$, then letting $\pi^j = (\gamma^j, \delta^j)$ we have $P'' = \llbracket \mathcal{A}, \gamma^j \rrbracket$.*

765 **PROOF.** We proceed by induction on the transition number $j \geq 0$.

base case $j = 1$: This case falls into the case of Proposition 16.

inductive case: We assume, by inductive hypothesis, that $\exists P'$ such that

$$P \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau}^{j-1} P'$$

and $P' = \llbracket \mathcal{A}, \gamma^{j-1} \rrbracket$. As P' is the encoding of an extended interactive process, by Proposition 16, it exists P'' such that $P' \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau} P''$, and $P'' = \llbracket \mathcal{A}, \gamma^j \rrbracket$.

770 **Proposition 18 (Completeness 1).** *Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ and $\pi^1 = (\gamma^1, \delta^1)$. Then, $P \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau} P' = \llbracket \mathcal{A}, \gamma^1 \rrbracket$.*

PROOF. By Proposition 16, if there exists P' such that $P \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau} P'$, then the structure of P' is deterministically computed.

Now, to prove that always exists P' , we observe that even in the case no reaction a is applicable in the interactive process π in A , then process P can always execute a step transition, as its subprocesses P_a can always execute one of the *alternative code for when reaction a is not applicable* (see Definition 11, code for P_a processes).

Corollary 19 (Completeness 2). *Let $P = \llbracket A, \gamma \rrbracket$ and $\pi^j = (\gamma^j, \delta^j)$. Then,*
 $P \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau} P'' = \llbracket A, \gamma^j \rrbracket$.

PROOF. The proof proceeds by induction on the number j , and it is similar to the one of Corollary 17.

Theorem 32 (Correspondence). $\sim_F = \equiv_{\mathcal{L}_F}$

PROOF. The proof is just an adaptation of the classical result. The two implications are proved separately.

$\sim_F \subseteq \equiv_{\mathcal{L}_F}$: Given any two processes $P \sim_F Q$ we need to prove that for any bioHML formula G we have $P \models G$ iff $Q \models G$. Without loss of generality, we prove that $P \models G$ implies $Q \models G$. The proof is by structural induction on G .

- if $G = \mathbf{t}$, then $Q \models G$.
- if $G = \mathbf{f}$, then the assumption $P \models G$ is false and the implication holds.
- if $G = G_1 \wedge G_2$ we take as inductive hypotheses that

$$\begin{aligned} \forall R, S. R \sim_F S \wedge R \models G_1 &\Rightarrow S \models G_1 \\ \forall R, S. R \sim_F S \wedge R \models G_2 &\Rightarrow S \models G_2 \end{aligned}$$

We need to prove that $Q \models G$. Since $P \models G = G_1 \wedge G_2$ we have $P \models G_1$ and $P \models G_2$. Since $P \sim_F Q$, by inductive hypotheses we get $Q \models G_1$ and $Q \models G_2$. Hence $Q \models G_1 \wedge G_2 = G$.

- if $G = G_1 \vee G_2$ we take as inductive hypotheses that

$$\begin{aligned} \forall R, S. R \sim_F S \wedge R \models G_1 &\Rightarrow S \models G_1 \\ \forall R, S. R \sim_F S \wedge R \models G_2 &\Rightarrow S \models G_2 \end{aligned}$$

We need to prove that $Q \models G$. Since $P \models G = G_1 \vee G_2$ we have $P \models G_1$ or $P \models G_2$. If $P \models G_1$, since $P \sim_F Q$, by inductive hypotheses we get $Q \models G_1$ and thus $Q \models G_1 \vee G_2 = G$. If $P \models G_2$, since $P \sim_F Q$, by inductive hypotheses we get $Q \models G_2$ and thus $Q \models G_1 \vee G_2 = G$.

- if $G = \langle \chi \rangle H$ we take as inductive hypothesis that

$$\forall R, S. R \sim_F S \wedge R \models H \Rightarrow S \models H$$

We need to prove that $Q \models G$. Since $P \models \langle \chi \rangle H$ it means that there exists v, P' such that $P \xrightarrow{v} P'$ with $v \models \chi$ and $P' \models H$. Since $P \sim_F Q$, there exists w, Q' such that $Q \xrightarrow{w} Q'$ with $w \models \chi$ and $P' \sim_F Q'$. Then, by inductive hypothesis, $Q' \models H$ and thus $Q \models \langle \chi \rangle H = G$.

- if $G = [\chi]H$ we take as inductive hypothesis that

$$\forall R, S. R \sim_F S \wedge R \models H \Rightarrow S \models H$$

We need to prove that $Q \models G$. If there is no $v \models \chi$ such that $Q \xrightarrow{v} Q'$ for some Q' , then $Q \models [\chi]H = G$ trivially. For any v, Q' such that $Q \xrightarrow{v} Q'$ with $v \models \chi$, then as $P \sim_F Q$ there must exist w, P' such that $P \xrightarrow{w} P'$ with $w \models \chi$ and $P' \sim_F Q'$. Since $P \models G = [\chi]H$ then it must be $P' \models H$. Since $P' \sim_F Q'$, by inductive hypothesis $Q' \models H$. Hence $Q \models [\chi]H = G$.

$\equiv_{\mathcal{L}_F} \subseteq \sim_F$: We prove that $\equiv_{\mathcal{L}_F}$ is a bio-simulation and thus included in \sim_F . Take two generic processes $P \equiv_{\mathcal{L}_F} Q$ and suppose $P \xrightarrow{v} P'$ for some v, P' .

- If $v \models F$ we want to prove that there exists some w, Q' such that $Q \xrightarrow{w} Q'$, with $w \models F$ and $P' \equiv_{\mathcal{L}_F} Q'$.

Towards a contradiction, assume that we cannot find such w, Q' .

If there is no transition $Q \xrightarrow{w} Q'$ such that $w \models F$, then the bioHML formula $G \triangleq \langle F \rangle \mathbf{t}$ is such that $P \models G$ and $Q \not\models G$, contradicting the assumption $P \equiv_{\mathcal{L}_F} Q$.

Otherwise, let $\mathcal{Q} \triangleq \{Q' \mid \exists w. Q \xrightarrow{w} Q' \wedge w \models F\}$ be the (non-empty) set of processes reachable from Q via a transition with a (complete) label that satisfies F . Since our processes are with guarded recursion, the set \mathcal{Q} is finite. Let $\mathcal{Q} = \{Q'_1, \dots, Q'_n\}$. By hypothesis all processes in \mathcal{Q} must not be bio-logically equivalent to P' , hence for any $i \in [1, n]$ there exists a bioHML formula G_i such that $P' \models G_i$ and $Q'_i \not\models G_i$ (if it was the opposite, $P' \not\models H_i$ and $Q'_i \models H_i$ for some H_i , we can use the converse formula $G_i \triangleq \overline{H_i}$). But then the formula $G \triangleq \langle F \rangle (G_1 \wedge \dots \wedge G_n)$ is such that $P \models G$ and $Q \not\models G$, contradicting the assumption $P \equiv_{\mathcal{L}_F} Q$.

- If $v \not\models F$ then the proof is analogous to the previous case (by exploiting $\neg F$) and thus omitted.

A process algebraic approach to reaction systems [☆]

Linda Brodo^b, Roberto Bruni^a, Moreno Falaschi^c

^a*Dipartimento di Informatica, Università di Pisa, Italy*

^b*Dipartimento di Scienze Economiche e Aziendali, Università di Sassari, Italy*

^c*Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, Univ. di Siena, Italy*

Abstract

In the area of Natural Computing, Reaction Systems (RSs) are a qualitative abstraction inspired by the functioning of living cells, suitable to model the main mechanisms of biochemical reactions. RSs interact with a context, and pose challenges for modularity, compositionality, extendibility and behavioural equivalence. In this paper we define a modular encoding of RSs as processes in the chained Core Network Algebra (cCNA), which is a new variant of the link-calculus. The encoding represents the behaviour of each entity separately and preserves faithfully their features, and we prove its correctness and completeness. Our encoding provides a Labelled Transition System (LTS) semantics for RSs. Based on the LTS semantics, we adapt the classical notion of bisimulation to define a novel equivalence, called bio-similarity, for studying properties of RSs. In particular, we define a new assertion language based on regular expressions, which allows us to specify the properties of interest, and use it to extend Hennessy-Milner logic to our setting. We prove that our bio-similarity relation and the logical equivalence, that are defined parametrically on some assertion of interest, coincide. Finally, we claim that our encoding contributes to increase the expressiveness of RSs, by exploiting the interaction among different RSs.

Keywords: process algebras, Reaction Systems, assertion language, HM-logic

1. Introduction

Natural Computing is an active area of research which builds on two main aspects: human designed computing inspired by nature, and computation performed in nature. Reaction Systems (RSs) [1] are a rewriting formalism inspired by the way biochemical reactions take place in living cells. This theory has already shown to be relevant in several different fields, including biology [2, 3, 4, 5]

[☆]Research partially supported by Università degli Studi di Sassari *Fondi di Ateneo per la ricerca 2019*.

Email addresses: `brodo@uniss.it` (Linda Brodo), `bruni@di.unipi.it` (Roberto Bruni), `moreno.falaschi@unisi.it` (Moreno Falaschi)

and molecular chemistry [6]. RSs formalise the mechanisms of biochemical systems, such as *facilitation* and *inhibition*. As a qualitative approximation of the real biochemical reactions, they consider if a necessary reagent is present or not, and likewise they consider if an inhibiting molecule is present or not. The possible reactants and inhibitors are called ‘entities’. RSs model in a direct way the interaction of a living cell with the environment (called ‘context’). However, two RSs are seen as independent models and do not interact.

In this paper, which is an extended version of [7], we present an encoding from RSs to the chained Core Network Algebra (cCNA), a variant of the open multiparty process algebra CNA [8]; the CNA equipped with mobility is referred to as the **link**-calculus [9, 10]. Here mobility is not needed. This formalism allows several processes to synchronise and communicate altogether, at the same time, with a new interaction mechanism based on links and link chains. The initial motivation for introducing an open multiparty mechanism in [9] was to encode Mobile Ambients [11], getting a much stronger operational correspondence than any available in the literature, such as the ones in [12, 13]. Later it was shown that the **link**-calculus allowed one to easily encode calculi for biology equipped with membranes, as in [14].

We illustrate our embedding by means of some examples coming both from the computer science and the biological field. We also show that our embedding preserves the main features of RSs, and prove its correctness and completeness from the operational semantics viewpoint.

Then, we present a methodology for verifying formally properties of Reaction Systems. The classical notion of bisimulation for process algebras allows to consider two processes as equivalent when one process can simulate all the actions executed by the other one and vice versa. In this paper we define a new notion of bisimilarity which takes into account the characteristics of biological systems. We define a new *bio-simulation* relation having in mind the possibility that two interacting systems may be compared w.r.t. a subset of the possible biological actions. This is useful for concentrating on the sub-model that one may need to consider for a specific study or application, without getting lost in the complexity of the full biological system, or network. The notion of bio-simulation relies on a new and simple assertion language, which allows to focus on some properties of the Reaction Systems to be verified. In fact, bio-similarity is parametric on a given assertion of interest. Then we prove that the well-known logical characterisation of bisimilarity in terms of Hennessy-Milner Logic [15] (HML) can be extended to the case of bio-similarity by tailoring HML formulas to the same assertion of interest used in bio-similarity. As HML is a powerful formalism for specifying properties of labeled transition systems, its variant introduced here is a suitable option to specify formulas over complex labels by abstracting from unnecessary details.

Our main contributions are as follows:

- RSs are encoded in a modular way, in the sense that all constituents of a RS, namely entities, reactions and the context, are seen as interacting cCNA processes; in principle, this allows to study the behaviour of any con-

stituent in isolation, contrary to what happens in the basic RS framework, where the evolution is possible only when all constituents are given;

- any context is represented as an ordinary cCNA process, which allows us to specify recursive and non-deterministic contexts in a natural way; when deterministic contexts are considered, as in ordinary RSs, the cCNA computation is also deterministic and matches the evolution of the underlying RS;
- we define an assertion language to specify local properties of RSs and exploit it to define a novel notion of behavioural equivalence, called bio-similarity;
- we show that our assertions can be used to extend the Hennessy-Milner's logic to our encoding, in such a way that logical equivalence of processes coincide with bio-similarity.

Moreover, we sketch how the encoding can be used to enhance the expressivity of RSs along different dimensions:

- we sketch how one can express the behaviour of entity mutation, in such a way that the mutated version of the entity s can take part to only a subset of reactions requiring entity s ;
- we sketch how with a little coding effort, our embedding allows two RSs to communicate; i.e. we can model scenarios where a subset of those entities that the context can provide, are instead provided by a second RS.

The main drawback of our proposal, is that the resulting cCNA code is verbose. Nevertheless it is clear that our encoding can be automatised by means of a proper front-end in an implementation of the `link`-calculus. The examples that we propose also show that some optimisations are possible to reduce the coding efforts when suitable assumptions are made about the provision of entities.

Related work. Process calculi have been used successfully to model biological processes, see [16] for a recent survey. We are not aware of any Structural Operational Semantics for RS; it seems not the case that a deterministic behaviour, as the one of the RS, once the initial state is fixed, could be defined by inference rules that classically are applied to define non-deterministic transition systems. The reversible computation paradigm for RS extends the RS framework by allowing backward computations as well as forward computations; for this goal a set of inference rules for rewriting logic has been defined in [17] to exactly keep trace of those elements that dissolve in the next computation step.

The proposed network of RSs [18] allows any RS in the network to receive entities produced from its neighbours (that will represent its context), with the hypothesis that RSs without neighbours will receive no entities from the context. In the idea we sketch, a RS can receive some entities from the context and some

others from a second RS; moreover we can represent contexts having a recursive, non-deterministic behaviour. By exploiting recursion, the kind of interactions which can be defined can be complex and expressive. Example 34 and more in general the discussion in Section 7.2 show that the interaction between RSs can help to model new scenarios.

As already mentioned, a preliminary version of this paper appeared in [7]. There are several major differences w.r.t. the conference version [7], which is here extended as follows:

- we present several new examples (those in Sections 5 and 6.3) to illustrate our encoding, and give more detailed explanations about its use;
- we introduce an assertion language to specify the properties of RSs;
- we define the notion of bio-simulation to relate RSs and compare their behaviours;
- we show that our assertion language allows to specify properties extending to our encoding the Hennessy-Milner logic;
- we include here all proofs of main results.

Structure of the paper. Section 2 describes RSs and their semantics (interactive processes). Section 3 briefly describes the cCNA process algebra and its operational semantics. Section 4 defines the embedding of RSs in cCNA processes and shows some simple examples to illustrate it. Section 5 presents a couple of examples with automata theory and biological applications. Section 6 presents a methodology for the formal verification of properties of Reaction Systems that are expressed in a novel assertion language. Section 7 presents some features and advantages of our embedding for the compositionality of RSs. Finally, Section 8 discusses future work, and concludes.

2. An Overview of Reaction Systems

Natural Computing is concerned with human-designed computing inspired by nature as well as with computation taking place in nature. The theory of Reaction Systems [1] was born in the field of Natural Computing to model the behaviour of biochemical reactions taking place in living cells. Despite its initial aim, this formalism has shown to be quite useful not only for modeling biological phenomena, but also for the contributions which is giving to computer science [19], theory of computing, mathematics, biology [2, 3, 4, 5], and molecular chemistry [6]. Here we briefly review the basic notions of RSs, see [1] for more details.

The mechanisms that are at the basis of biochemical reactions and thus regulate the functioning of a living cell, are *facilitation* and *inhibition*. These mechanisms are reflected in the basic definitions of RSs.

Definition 1 (Reaction). Let S be a set of entities. A reaction over S is a triple $a = (R, I, P)$, where R, I, P are finite, non empty subsets of S and $R \cap I = \emptyset$.

The sets R, I, P are also written R_a, I_a, P_a and called the *reactant set* of a , the *inhibitor set* of a , and the *product set* of a , respectively. All reactants are needed for the reaction to take place. Any inhibitor blocks the reaction if it is present. Products are the outcome of the reaction. Also, $R_a \cup I_a$ is the set of the resources of a and $rac(S)$ denotes the set of all reactions in S . Because R and I are non empty, all products are produced from at least one reactant and every reaction can be inhibited in some way. Sometimes artificial inhibitors are used that are never produced by any reaction. For the sake of simplicity, in some examples, we will allow I to be empty.

Definition 2 (Reaction System). A Reaction System (RS) is an ordered pair $\mathcal{A} = (S, A)$ such that S is a finite set of entities, and $A \subseteq rac(S)$ is a set of reactions over S .

The set S is called the *background set* of \mathcal{A} ; its elements represent molecular substances (e.g., atoms, ions, molecules) that may be present in the states of a biochemical system. The set A is the set of *reactions* of \mathcal{A} . Since S is finite, so is A : we denote by $|A|$ the number of reactions in A .

Definition 3 (Reaction Result). Given a set of entities S , let $W \subseteq S$ be a finite subset of entities.

1. Let $a \in rac(S)$ be a reaction over S . Then a is enabled by the entities in the set W , denoted by $en_a(W)$, if $R_a \subseteq W$ and $I_a \cap W = \emptyset$, i.e. all the reactants of a are in W , while none of the inhibitors of a are in W . The result of a on W , denoted by $res_a(W)$, is defined by: $res_a(W) \triangleq P_a$, if $en_a(W)$, and $res_a(W) \triangleq \emptyset$ otherwise.
2. Let $A \subseteq rac(S)$ be a (finite) set of reactions over S . The result of A on W , denoted by $res_A(W)$, is defined by: $res_A(W) \triangleq \bigcup_{a \in A} res_a(W)$.

The theory of Reaction Systems is based on the following assumptions.

- **No permanency.** An entity of a set W vanishes unless it is sustained by a reaction. This reflects the fact that a living cell would die for lack of energy, without chemical reactions.
- **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 taken into account.
- **Threshold nature of resources.** From the previous item, we assume that either an entity is available and there is enough of it (i.e. there are no conflicts), or it is not available at all.

The dynamic behaviour of a RS is formalized in terms of *interactive processes*.

Definition 4 (Interactive Process). Let $\mathcal{A} = (S, A)$ be a RS and let n be a nonnegative integer; An n -step *interactive process* in \mathcal{A} is a pair $\pi = (\gamma, \delta)$ of finite sequences s.t. $\gamma = \{C_i\}_{i \in [0, n]}$ and $\delta = \{D_i\}_{i \in [0, n]}$ where $C_i, D_i \subseteq S$ are set of entities for any $i \in [0, n]$, $D_0 = \emptyset$, and $D_i = \text{res}_A(D_{i-1} \cup C_{i-1})$ for any $i \in [1, n]$.

Living cells are seen as open systems that continuously react with the external environment, in discrete steps. The sequence γ is the *context sequence* of π ; it can be arbitrarily defined and represents the influence of the environment on the RS. The sequence δ is the *result sequence* of π and it is entirely determined by γ and A . The sequence $\tau = W_0, \dots, W_n$ with $W_i = C_i \cup D_i$, for any $i \in [0, n]$ is called a *state sequence*. Each state W_i in a state sequence is the union of two sets: the context C_i at step i and the result $D_i = \text{res}_A(W_{i-1})$ from the previous step.

Since we will be able to deal with recursively contexts, we extend the notion of an interactive process to deal with infinite sequences.

Definition 5 (Extended Interactive Process). Let $\mathcal{A} = (S, A)$ be a RS, and let $\pi = (\gamma, \delta)$ be an n -step interactive process, with $\gamma = \{C_i\}_{i \in [0, n]}$ and $\delta = \{D_i\}_{i \in [0, n]}$. Then, we let $\pi^\infty = (\gamma^\infty, \delta^\infty)$ be the extended interactive process of π , defined as $\gamma^\infty = \{C'_i\}_{i \in \mathbb{N}}$, $\delta^\infty = \{D'_i\}_{i \in \mathbb{N}}$, where:

$$C'_j = \begin{cases} C_j & \text{if } j \in [0, n] \\ \emptyset & \text{if } j > n \end{cases} \quad D'_j = \begin{cases} D_0 & \text{if } j = 0 \\ \text{res}_A(D'_{j-1} \cup C'_{j-1}) & \text{if } (j \geq 1) \end{cases}$$

Given an extended interactive process $\pi = (\gamma, \delta)$, we denote by π^k the shift of π starting at the k -th state sequence; formally we let $\pi^k = (\gamma^k, \delta^k)$ with $\gamma^k = \{C'_i\}_{i \in \mathbb{N}}$, $\delta^k = \{D'_i\}_{i \in \mathbb{N}}$ with $C'_0 = C_k \cup D_k$, $D'_0 = \emptyset$, and $C'_i = C_{i+k}$, $D'_i = D_{i+k}$ for any $i \geq 1$.

3. Chained CNA (cCNA)

In this section we introduce the syntax and operational semantics of the process algebra cCNA (chained CNA) [7] to be used for encoding RSs. As already explained in the Introduction, cCNA is a variant of CNA [8], the non-mobile fragment of **link**-calculus [9, 10]. In cCNA the action prefixes are link chains and not just links.

Link Chains. Let \mathcal{C} be the set of channels, ranged over by $\mathbf{a}, \mathbf{b}, \dots$, and let $\text{Act} \triangleq \mathcal{C} \cup \{\tau\} \cup \{\square\}$ be the set of actions, ranged over by α, β, \dots , where the symbol τ denotes a *silent* action, while the symbol \square denotes a *virtual* (non-specified) action. A *link* is a pair $\ell = \alpha \backslash \beta$; it is *solid* if $\alpha, \beta \neq \square$; intuitively, α and β are two interaction points, one for incoming requests and the other for outgoing requests. The link $\square \backslash \square$ is called *virtual*. A link is *valid* if it is solid or virtual. We let \mathcal{L} be the set of valid links. A *link chain* is a finite sequence $v = \ell_1 \dots \ell_n$ of (valid) links $\ell_i = \alpha_i \backslash \beta_i$ such that:

1. for any $i \in [1, n-1]$, $\begin{cases} \beta_i, \alpha_{i+1} \in \mathcal{C} & \text{implies } \beta_i = \alpha_{i+1} \\ \beta_i = \tau & \text{iff } \alpha_{i+1} = \tau \end{cases}$
2. $\exists i \in [1, n]. \ell_i \neq \square \backslash \square$.

Virtual links represent missing elements of a chain. A chain is called *solid* if it does not contain any virtual link. The empty chain is denoted by ϵ . The equivalence \blacktriangleleft models expansion/contraction of virtual links to adjust the length of a link chain.

Definition 6 (Equivalence \blacktriangleleft). We let \blacktriangleleft be the least equivalence relation over link chains closed under the axioms (whenever both sides are well defined):

$$\begin{array}{ll} v \square \backslash \square & \blacktriangleleft v \\ \square \backslash \square v & \blacktriangleleft v \\ v_1 \square \backslash \square \backslash \square v_2 & \blacktriangleleft v_1 \square \backslash \square v_2 \\ v_1 \alpha \backslash \alpha \backslash \beta v_2 & \blacktriangleleft v_1 \alpha \backslash \alpha \backslash \beta v_2 \end{array}$$

Two link chains of equal length can be merged whenever each position occupied by a solid link in one chain is occupied by a virtual link in the other chain and solid links in adjacent positions match. Positions occupied by virtual links in both chains remain virtual. Merging is denoted by $v_1 \bullet v_2$. For example, given $v_1 = \tau \backslash \alpha \backslash \square \backslash \square$, $v_2 = \square \backslash \alpha \backslash \square \backslash \square$ and $v = \tau \backslash \alpha \backslash \square \backslash \square$ we have $v_1 \bullet v_2 = v$, whereas $v_1 \bullet v$ is not defined. Notably the merge operation is commutative and associative.

Some names in a link chain can be restricted as non observable and transformed into silent actions τ . This is possible only if they are matched by some adjacent link. Restriction is denoted by $(\nu a)v$. For example, given $v = \tau \backslash \alpha \backslash \square \backslash \square$ as above, we have $(\nu a)v = \tau \backslash \tau \backslash \square \backslash \square$, whereas $(\nu b)v$ is not defined.

Syntax. The set of cCNA processes, denoted as \mathcal{P} and ranged over by P, Q , is defined by the following grammar:

$$P, Q ::= \mathbf{0} \mid v.P \mid P + Q \mid P|Q \mid (\nu a)P \mid A$$

where v_i is a link chain, and A is a process identifier. The syntax of cCNA extends that of CNA [8] by allowing to use link chains as prefixes instead of links, i.e. we allow to write $v.P$ instead of $\ell.P$. For the rest it features nondeterministic choice $P + Q$ (also called sum), parallel composition $P|Q$, restriction $(\nu a)P$, and possibly recursively defined process identifiers A . Here we do not consider name mobility, which is present instead in the **link**-calculus and we omit the relabelling operator of CNA (not needed in the encoding).

As common in process algebras we restrict to consider prefix-guarded sums $v_1.P_1 + v_2.P_2$ and, exploiting associativity, we use the shorthand $\sum_{i \in I} v_i.P_i$ for a finite set of indexes $I = \{i_1, \dots, i_k\}$ instead of $v_{i_1}.P_{i_1} + \dots + v_{i_k}.P_{i_k}$. The inactive process $\mathbf{0}$ is thus just the empty summation.

In a prefix $v.P$ we can always assume that the links at the extremities of v are solid: if v needs to be used in larger chains, the operational semantics will add as many virtual links as needed by exploiting the equivalence \blacktriangleleft (see rule

$$\begin{array}{c}
\frac{v \blacktriangleright v_j \quad j \in I}{\sum_{i \in I} v_i.P_i \xrightarrow{v} P_j} \text{ (Sum)} \quad \frac{P \xrightarrow{v} P' \quad (A \triangleq P) \in \Delta}{A \xrightarrow{v} P'} \text{ (Ide)} \\
\\
\frac{P \xrightarrow{v} P'}{(\nu a)P \xrightarrow{(\nu a)v} (\nu a)P'} \text{ (Res)} \quad \frac{P \xrightarrow{v} P'}{P|Q \xrightarrow{v} P'|Q} \text{ (Lpar)} \quad \frac{Q \xrightarrow{v} Q'}{P|Q \xrightarrow{v} P|Q'} \text{ (Rpar)} \\
\\
\frac{P \xrightarrow{v'} P' \quad Q \xrightarrow{v} Q'}{P|Q \xrightarrow{v \bullet v'} P'|Q'} \text{ (Com)}
\end{array}$$

Figure 1: SOS semantics of cCNA processes.

Sum in Fig. 1). For example, the process $a \backslash_b.P$ and $a \backslash_b \square \backslash_{\square}.P$ are completely equivalent.

Regarding process constants, we rely on a given set $\Delta = \{A_i \triangleq P_i\}_{i \in I}$ of (possibly recursive) process definitions.

Semantics. The operational semantics of cCNA is defined in the SOS style by the inference rules in Fig.1. The rules are reminiscent of those for Milner's CCS and they essentially coincide with those of CNA in [8]. The only difference is due to the presence of prefixes that are link chains. Briefly: rule (*Sum*) selects one alternative and puts as label a possible contraction/expansion of the link chain in the selected prefix; rule (*Ide*) selects one transition of the defining process for a constant; rule (*Res*) restricts some names in the label (it cannot be applied when $(\nu a)v$ is not defined); rules (*Lpar*) and (*Rpar*) account for interleaving in parallel composition; rule (*Com*) synchronises interactions (it cannot be applied when $v \bullet v'$ is not defined).

Example 7. As some simple examples, consider the recursive process definitions $H \triangleq a \backslash_b. a \backslash_c.H$ and $K \triangleq a \backslash_b.K + a \backslash_c.K$: the former recursively provides a link from a to b and then, at the next step, from a to c ; the latter provides at each step a link from a to b or from a to c , nondeterministically.

Analogously to CNA, the operational semantics of cCNA satisfies the so called Accordion Lemma: whenever $P \xrightarrow{v} P'$ and $v' \blacktriangleright v$ then $P \xrightarrow{v'} P'$.

3.1. Notation for link chains

Hereafter we make use of some new notations for link chains that will facilitate the presentation of our encoding.

Definition 8 (Replication). Let v be a valid link chain such that vv is also a valid link chain. The n times replication of v , written v^n , is defined recursively by letting $v^0 = \epsilon$ (i.e. the empty chain) and $v^n = vv^{n-1}$.

For example, the expression $(a \backslash_b^{\square} \backslash_{\square})^3$ denotes the chain $a \backslash_b^{\square} \backslash_{\square}^a \backslash_b^{\square} \backslash_{\square}^a \backslash_b^{\square} \backslash_{\square}$. Instead the expression $(a \backslash_b)^2$ is ill-defined because a does not match with b .

Then, we introduce the notation for *half links* that will be used in conjunction with the *open block of chain* to form regular link chains.

Definition 9 (Half links). Let a be a channel name, we denote by $a \backslash$ the *half left link*, and by \backslash_a the *half right link*.

In the encoding of RS we will make extensive use of subscripted names: each name a will come in two variants a_i and a_o . This is just a technical issue to prevent accidental matching between links. To see why this is important, compare the chain $\tau \backslash_a^{\square} \backslash_{\square}^a \backslash_{\tau}$ with $\tau \backslash_{a_i}^{\square} \backslash_{\square}^{a_o} \backslash_{\tau}$: the former is \blacktriangleleft equivalent to the solid chain $\tau \backslash_a^a \backslash_{\tau}$; the latter cannot become solid unless merged with a chain that links a_i to a_o , like $\square \backslash_{\square}^{a_i} \backslash_{a_o}^{\square} \backslash_{\square}$. This form of subscripting is exploited in the definition of open blocks.

Definition 10 (Open block). Let σ be a finite sequence of names. We define an *open block* as $(\bigvee_{a \in \sigma} \square \backslash_{a_i}^{\square} \backslash_{\square}^{a_o})$, where a_i and a_o are annotated version of the name a (as explained above), by letting

$$\left(\bigvee_{a \in \sigma} \square \backslash_{a_i}^{\square} \backslash_{\square}^{a_o} \right) \triangleq \begin{cases} \epsilon & \text{if } \sigma = \epsilon \text{ is the empty sequence} \\ \square \backslash_{b_i}^{\square} \backslash_{\square}^{b_o} \left(\bigvee_{a \in \rho} \square \backslash_{a_i}^{\square} \backslash_{\square}^{a_o} \right) & \text{if } \sigma = b\rho \end{cases}$$

Abusing the notation, we will use the open block notation for sets of names rather than sequences, assuming the names in the set are taken according to some default order (e.g. the lexicographic one).

We then combine half links and open blocks to form valid link chains.

For example, for $X = \{a, b\}$ the expression $(\bigvee_{c \in X} \square \backslash_{c_i}^{\square} \backslash_{\square}^{c_o})$ denotes the block $\square \backslash_{a_i}^{\square} \backslash_{\square}^{a_o} \backslash_{b_i}^{\square} \backslash_{\square}^{b_o}$; and the expression $r_1 \backslash \left(\bigvee_{c \in X} \square \backslash_{c_i}^{\square} \backslash_{\square}^{c_o} \right) \backslash_{r_2}$ denotes the chain $r_1 \backslash_{a_i}^{\square} \backslash_{\square}^{a_o} \backslash_{b_i}^{\square} \backslash_{\square}^{b_o} \backslash_{r_2}$.

4. From Reaction Systems to cCNA

Here we define an encoding of Reaction Systems into cCNA. The idea is to define separated processes for representing the behaviour of each entity, each reaction, and for the provisioning of each entity by the context.

In the following we refer to a given set of entities S and a set of reactions $A \subseteq \text{rac}(S)$, i.e. that the Reaction System $\mathcal{A} = (S, A)$ is known.

Processes for entities. Given an entity $s \in S$, we exploit five different pairs of channel names for the interactions over s :

- names s_i, s_o are used to test the presence of s in the system;
- names $\widehat{s}_i, \widehat{s}_o$ are used to test the provisioning of s from the context;

- names \tilde{s}_i, \tilde{s}_o are used to test the production of s by some reaction;
- names \bar{s}_i, \bar{s}_o are used to test the absence of s in the system;
- names $\underline{s}_i, \underline{s}_o$ are used to test the absence of s from the context.

We let P_s be the process implementing the presence of s in the system, and $\overline{P_s}$ be the one for its absence. They can be seen as instances of the same template, which is given below.

$$\begin{aligned}
P_s &\triangleq E(s, \tilde{s}, \hat{s}, \underline{s}) & \overline{P_s} &\triangleq E(\bar{s}, \tilde{s}, \hat{s}, \underline{s}) \\
E(s, \tilde{s}, \hat{s}, \underline{s}) &\triangleq \sum_{h,k \geq 0} (s_i \setminus_{s_o} \square)^h \hat{s}_i \setminus_{\hat{s}_o} \square (\tilde{s}_i \setminus_{\tilde{s}_o} \square)^k . P_s \\
&\quad + \sum_{h \geq 0, k \geq 1} (s_i \setminus_{s_o} \square)^h \underline{s}_i \setminus_{\underline{s}_o} \square (\tilde{s}_i \setminus_{\tilde{s}_o} \square)^k . P_s \\
&\quad + \sum_{h \geq 0} (s_i \setminus_{s_o} \square)^h \underline{s}_i \setminus_{\underline{s}_o} . \overline{P_s}
\end{aligned}$$

The first line of $E(s, \tilde{s}, \hat{s}, \underline{s})$ accounts for the case where s is tested for presence by h reactions and produced by k reactions, while being provided by the context $(\hat{s}_i \setminus_{\hat{s}_o})$. Thus, s will be present at the next step (the continuation is P_s). Here h and k are not known a priori and therefore any combination is possible.

In practice, by knowing the number of reactions that test s , we can bound the maximum values of h and k . The second line accounts for the analogous case where s is not provided by the context $(\underline{s}_i \setminus_{\underline{s}_o})$. The condition $k \geq 1$ guarantees that s will remain present (the continuation is P_s). The third line accounts for the case where s is tested for presence, but it is neither produced nor provided by the context. Therefore, in the next step s will be absent in the system (the continuation is $\overline{P_s}$). Note that in the case of $\overline{P_s}$ the test for presence of s in the system is just replaced by the test for its absence.

Processes for reactions. Here we focus on the encoding of the set of reactions $A \subseteq \text{rac}(S)$. We assume that all the reactions a are numbered and use j as an index for reactions. We introduce two channel names for each reaction aj :

- r_j to mark the occurrence of the reaction;
- p_j to mark the product set of the reaction.

We shall exploit names r_j, p_j to join the chains provided by the application of all the reactions. The process for the j th reaction $aj = (R_j, I_j, P_j)$ must assert either the possibility to apply the reaction or its impossibility. The first case happens when all its reactants are present (the link $s_i \setminus_{s_o}$ is requested for any $s \in R_j$) and all its inhibitors are absent (the link $\bar{e}_i \setminus_{\bar{e}_o}$ is requested for any $e \in I_j$), then the product set is released (the link $\tilde{c}_i \setminus_{\tilde{c}_o}$ is requested for any $c \in P_j$). The second case can happen for two reasons: one of the reactants is absent (the link $\bar{s}_i \setminus_{\bar{s}_o}$ is requested for some $s \in R_j$) or one of the inhibitors is

present (the link $e_i \setminus_{e_o}$ is requested for some $e \in I_j$). The process is recursive so that reactions can be applied at any step.

$$\begin{aligned}
P_{aj} &\triangleq \\
&r_j \setminus \left(\left(\bigsqcup_{s \in R_j} \square \setminus_{s_i} s_o \right) \setminus \left(\bigsqcup_{e \in I_j} \square \setminus_{\bar{e}_i} \bar{e}_o \right) \setminus_{r_{j+1}} \square \setminus_{p_j} \left(\bigsqcup_{c \in P_j} \square \setminus_{\tilde{c}_i} \tilde{c}_o \right) \setminus_{p_{j+1}} \right) . P_{aj} \quad \{aj \text{ is applicable}\} \\
&+ \\
&\sum_{s \in R_j} r_j \setminus \square \setminus_{\bar{s}_i} \bar{s}_o \setminus_{r_{j+1}} \square \setminus_{p_j} \setminus_{p_{j+1}} . P_{aj} \quad \{aj \text{ is not applicable}\} \\
&+ \\
&\sum_{e \in I_j} r_j \setminus \square \setminus_{e_i} e_o \setminus_{r_{j+1}} \square \setminus_{p_j} \setminus_{p_{j+1}} . P_{aj} \quad \{aj \text{ is not applicable}\}
\end{aligned}$$

Channels r_j and r_{j+1} enclose the enabling/disabling condition of reaction aj . Channels p_j and p_{j+1} enclose the links related to the entities produced by aj . Each reaction defines a pattern to be satisfied, i.e. each reaction inserts as many virtual links as the number of reactants, inhibitors, and products, as required by the corresponding reaction.

We will see that all the link chain labels of transitions follow the same schema: first we find all the reactions limited to the reactants and inhibitors (chained using r_j channels), then all the supplies by the contexts (chained using the channel cxt , to be introduced next), and finally the products for all the reactions (chained using p_j channels). For notational convenience, we fix that $r_{|A|+1} = cxt$ and $p_{|A|+1} = \tau$. This schema will be later illustrated in detail in Example 15.

Processes for contexts. For marking the part of the chain provided by the context, we exploit the name cxt . In RSs, the context sequence γ provides a set of entities C_n at each instant of time n : for each entity $s \in S$, the context must say if the entity is provided or not. Correspondingly, we introduce another process Cxt_n defined as follows:

$$Cxt_n \triangleq cxt \setminus \left(\bigsqcup_{s \in C_n} \square \setminus_{\widehat{s}_i} \widehat{s}_o \right) \setminus \left(\bigsqcup_{e \notin C_n} \square \setminus_{\underline{e}_i} \underline{e}_o \right) \setminus_{p_1} . Cxt_{n+1}$$

We only consider Cxt_n with $n > 0$, as the entities that are present at step zero are considered to be present in the initial system (if $s \in C_0$ the process P_s will be present initially, otherwise \bar{P}_s will be present).

Encoding. In the following we use the following conventions for denoting different categories of names:

- $decs \triangleq \{s, \bar{s}, \tilde{s}, \widehat{s}, \underline{s} \mid s \in S\}$ is the set of channel names for decorated entities (without subscripts $_i$ and $_o$);
- $ents \triangleq \{d_i, d_o \mid d \in decs\}$ is the set of channel names for entities;
- $reacts \triangleq \{r_1, \dots, r_{|A|+1}\}$ is the set of channel names r_j associated with each reaction aj (we remind that $r_{|A|+1} = cxt$);

- $prods \triangleq \{p_1, \dots, p_{|A|}\}$ is the set of channel names p_j for product sets associated with each reaction aj (we remind that $p_{|A|+1} = \tau$).

Definition 11 (Encoding). Let $\mathcal{A} = (S, A)$ be a RS, and let $\pi = (\gamma, \delta)$ be an extended interactive process in \mathcal{A} , with $\gamma = \{C_i\}_{i \in \mathbb{N}}$. We define its cCNA encoding $\llbracket \mathcal{A}, \gamma \rrbracket$ as follows:

$$\llbracket \mathcal{A}, \gamma \rrbracket \triangleq (\nu \text{ names}) \left(I \mid \prod_{a \in A} P_a \mid Cxt_1 \mid \prod_{s \in C_0} P_s \mid \prod_{s \notin C_0} \overline{P}_s \right)$$

where $\text{names} = \text{reacts} \cup \text{ents} \cup \text{prods} \cup \{cxt\}$. For technical reasons, we introduce the (trivially recursive) init process $I \triangleq \tau_{\setminus r_1}.I$: it is needed to allow the name r_1 to be matched at the start of any chain at any instant of time.

It is important to observe that, for each transition, our cCNA encoding requires all the processes running in parallel to interact in that transition. This is due to the fact that all the channel names r_j , p_j , cxt , including those for decorated names s_i , s_o , \overline{s}_i , \overline{s}_o , ... are restricted.

Lemma 12. Let $\mathcal{A} = (S, A)$ be a RS and let $\pi = (\gamma, \delta)$ be an extended interactive process in \mathcal{A} . Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ its cCNA encoding. If exists P' such that $P \xrightarrow{(\nu \text{ names})v} P'$ is a transition of P , then

1. for each reaction $aj \in A$, the corresponding channels r_j and p_j appear in v ; for each entity $s \in S$, the corresponding channel s (suitably decorated) appear in v ; the channel cxt appears in v ;
2. for each reaction $aj \in A$ and each virtual link offered by processes P_a and Cxt_1 is overlapped by exactly one solid link offered by processes representing entities.

The topmost restriction $(\nu \text{ names})$ appearing in the process $\llbracket \mathcal{A}, \gamma \rrbracket$ serves to guarantee that all names appearing in a link of the chain labelling a transition are matched. Since all names appearing in any prefix of $\llbracket \mathcal{A}, \gamma \rrbracket$ are restricted, in the transition $\llbracket \mathcal{A}, \gamma \rrbracket \xrightarrow{(\nu \text{ names})v} P'$ it means that the observation $(\nu \text{ names})v$ has the form $\tau_{\setminus \tau} \dots \tau_{\setminus \tau}$, i.e., it is silent, and that v is solid. Later on we will be interested in reasoning about the actual chain v used in the transition. It has the peculiarity to start and end with silent actions and to include all names in $\text{reacts} \cup \text{prods} \cup \{cxt\}$. As a matter of notation we call such chain v *complete*.

Definition 13 (Complete Chain). A chain v is called *complete* if it is solid (i.e., it contains no virtual link) and has silent actions τ at its extremes. We write $P \xrightarrow{v} P'$ to mean that $P \xrightarrow{v} P'$ with v complete.

We will then use $\langle \mathcal{A}, \gamma \rangle$ to refer to the encoding without topmost name restrictions, i.e.,

$$\langle \mathcal{A}, \gamma \rangle \triangleq I \mid \prod_{a \in A} P_a \mid Cxt_1 \mid \prod_{s \in C_0} P_s \mid \prod_{s \notin C_0} \overline{P}_s.$$

and we will focus on the complete transitions of $\langle \mathcal{A}, \gamma \rangle$.
 The following Corollary immediately follows from Lemma 12.

Corollary 14. *Let $\mathcal{A} = (S, A)$ be a RS and let $\pi = (\gamma, \delta)$ be an extended interactive process in \mathcal{A} . Let $Q = \langle \mathcal{A}, \gamma \rangle$ and $P = \llbracket \mathcal{A}, \gamma \rrbracket = (\nu \text{ names})Q$. Then $P \xrightarrow{(\nu \text{ names})v} P'$ iff $Q \xrightarrow{v} Q'$ and $P' = (\nu \text{ names})Q'$.*

Example 15. Let \mathcal{A} be a RS whose specification contains two entities, $s1$ and $s2$, and the reactions $r_1 = (s1, \emptyset, s2)$ and $r_2 = (s2, \emptyset, s1)$ that produce $s2$ if $s1$ is present and $s1$ if $s2$ is present. For simplicity, we consider empty sets of inhibitors, which are not allowed by Definition 1, but the reader can assume a void inhibitor is present in both reactions. Then, we assume an extended interactive process $\pi = (\gamma, \delta)$ where the context γ provides $s1$ and $s2$ at every step, but we assume that only $s1$ is initially present. Since the context sequence is constant, we omit the subscript from Cxt . The corresponding cCNA process is $\llbracket \mathcal{A}, \gamma \rrbracket \triangleq (\nu \text{ names})\langle \mathcal{A}, \gamma \rangle$, with

$$\langle \mathcal{A}, \gamma \rangle \triangleq I \mid P_{s1} \mid \overline{P}_{s2} \mid P_{r1} \mid P_{r2} \mid Cxt$$

where:

$$\begin{aligned} P_{r1} &\triangleq r_1 \setminus_{s1_i} \square \setminus_{s1_o} \square \setminus_{r2} \square \setminus_{s2_i} \square \setminus_{s2_o} \square \setminus_{p2} \cdot P_{r1} + \dots; \\ P_{r2} &\triangleq \dots + r_2 \setminus_{s2_i} \square \setminus_{s2_o} \square \setminus_{cxt} \square \setminus_{p2} \cdot P_{r2} + \dots; \\ P_{s1} &\triangleq s1_i \setminus_{s1_o} \square \setminus_{s1_o} \square \cdot P_{s1} + \dots; \\ \overline{P}_{s2} &\triangleq \overline{s2_i} \setminus_{s2_o} \square \setminus_{s2_o} \square \setminus_{s2_o} \square \cdot P_{s2} + \dots; \\ Cxt &\triangleq cxt \setminus_{s1_i} \square \setminus_{s1_o} \square \setminus_{s2_i} \square \setminus_{s2_o} \square \cdot Cxt \end{aligned}$$

For clarity of exposition, we show the code of the processes just in part, to focus on the prefixes that will be involved in the first transition of the system. In Figure 2 we show the structure of a link chain label related to the execution of such a transition. The yellow blocks are referred to init process I , to the processes encoding the reactions, P_{r1} and P_{r2} , and to the context Cxt . As the figure puts in evidence, these two kinds of processes determine the structure of the link chain, from end to end, i.e. from the left τ to the right one. We could say that these processes form the *backbone* of the interaction. In contrast, the processes encoding the entities, P_{s1} , \overline{P}_{s2} , provide the solid links to be merged with the virtual links of the backbone (i.e. to be plugged in the backbone). In Figure 2, at the bottom of the chain, we have underlined with brackets the origin of the solid links that appear in the chain: the notation $P_1(P_2, P_3)$ means that the segment of the link chain is delimited by the process P_1 and it leaves "holes" where processes between the brackets, in this case P_2 and P_3 , insert their links. Formally, we have

$$\langle \mathcal{A}, \gamma \rangle \xrightarrow{\tau \setminus_{r1} \setminus_{s1_i} \setminus_{s1_o} \setminus_{r2} \setminus_{s2_i} \setminus_{s2_o} \setminus_{cxt} \setminus_{s1_i} \setminus_{s1_o} \setminus_{s2_i} \setminus_{s2_o} \setminus_{p1} \setminus_{s2_i} \setminus_{s2_o} \setminus_{p2} \setminus_{\tau}} P'$$

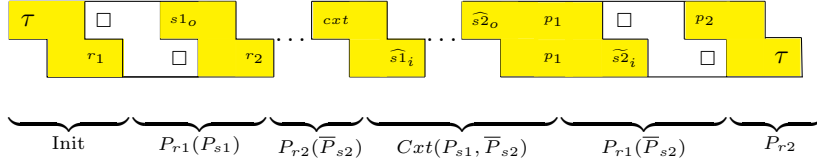


Figure 2: The link chain structure arising from reactions and context processes.

Since the chain in the transition label is complete we can also write

$$(\mathcal{A}, \gamma) \xRightarrow{\tau \setminus r_1 \setminus s_{1_i} \setminus s_{1_o} \setminus r_2 \setminus \bar{s}_{2_i} \setminus \bar{s}_{2_o} \setminus cxt \setminus \hat{s}_{1_i} \setminus \hat{s}_{1_o} \setminus \hat{s}_{2_i} \setminus \hat{s}_{2_o} \setminus p_1 \setminus \tilde{s}_{2_i} \setminus \tilde{s}_{2_o} \setminus p_2 \setminus \tau} P'$$

Example 15 outlines two different roles of the processes defining the translation of an interactive process: those processes encoding the reactions and the context provide the backbone of each transition, whereas the processes encoding the entities provide the resources needed for the communication to take place.

The flat function. Our transition labels are quite verbose; then, to simplify their processing, we introduce a function that takes a solid link chain and returns a simple string by eliminating all the channel matching pairs and leaving just one placeholder for them. This transformation is harmless, in the sense that it retains all the information in the chain, because it is applied to complete chains only. The function $flat(\cdot)$ is defined inductively as follows:

$$flat(\epsilon) \triangleq \epsilon \quad flat(\alpha \setminus \beta) \triangleq \begin{cases} \beta & \text{if } \beta \in reacts \cup \{cxt\} \cup prods \\ d & \text{if } \beta = d_i \text{ with } d \in decs \\ \epsilon & \text{otherwise} \end{cases}$$

$$flat(\alpha \setminus_\beta v) \triangleq flat(\alpha \setminus_\beta) flat(v)$$

where the usual string concatenation is represented by juxtaposition.

For example, if we consider again the complete label

$$v = \tau \setminus r_1 \setminus s_{1_i} \setminus s_{1_o} \setminus r_2 \setminus \bar{s}_{2_i} \setminus \bar{s}_{2_o} \setminus cxt \setminus \hat{s}_{1_i} \setminus \hat{s}_{1_o} \setminus \hat{s}_{2_i} \setminus \hat{s}_{2_o} \setminus p_1 \setminus \tilde{s}_{2_i} \setminus \tilde{s}_{2_o} \setminus p_2 \setminus \tau$$

from Example 15, we have

$$flat(v) = r_1 \ s1 \ r2 \ \bar{s}2 \ cxt \ \hat{s}1 \ \hat{s}2 \ p1 \ \tilde{s}2 \ p2.$$

It is then immediate to define the function $unflat$ to rebuild the complete label from the compact string (here we exploit again the half link and block notation):

$$unflat(x) \triangleq \begin{cases} x & \text{if } x \in reacts \cup \{cxt\} \cup prods \\ x_i \setminus x_o & \text{if } x \in decs \end{cases}$$

$$unflat(x_1 \dots x_n) \triangleq \tau \backslash unflat(x_1) \backslash \dots \backslash unflat(x_n) \backslash \tau$$

It is immediate to check that for any complete label v of our processes we have $v = unflat(flat(v))$.

With the next proposition, we analyse the structure of a cCNA process encoding of a reactive process after one transition step. In the following four statements, for brevity, we let $\mathcal{A} = (S, A)$ be a RS, and let $\pi = (\gamma, \delta)$ be an extended interactive process in A , with $\gamma = \{C_i\}_{i \in \mathbb{N}}$ and $\delta = \{D_i\}_{i \in \mathbb{N}}$.

Proposition 16 (Correctness 1). *Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ with*

$$P = (\nu \text{ names}) \left(I \mid \prod_{a \in A} P_a \mid Cxt_1 \mid \prod_{s \in C_0} P_s \mid \prod_{s \notin C_0} \bar{P}_s \right).$$

If there exists P' such that $P \xrightarrow{v} P'$, it holds that:

1. $v = \tau \backslash \dots \tau \backslash \tau$, and
2. $P' = (\nu \text{ names}) (I \mid \prod_{a \in A} P_a \mid Cxt_2 \mid \prod_{s \in C_1 \cup D_1} P_s \mid \prod_{s \notin C_1 \cup D_1} \bar{P}_s)$.

Moreover, given $\pi^1 = (\gamma^1, \delta^1)$, we have $P' = \llbracket \mathcal{A}, \gamma^1 \rrbracket$.

Now, we extend the previous result to a series of transitions.

Corollary 17 (Correctness 2). *Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ and $j \geq 1$. If there exists P'' such that $P \xrightarrow{\tau \backslash \dots \tau \backslash \tau}^j P''$, then letting $\pi^j = (\gamma^j, \delta^j)$ we have $P'' = \llbracket \mathcal{A}, \gamma^j \rrbracket$.*

With the following propositions, we prove that, given a RS $\mathcal{A} = (S, A)$ and an extended interactive process $\pi = (\gamma, \delta)$, then the cCNA process $\llbracket \mathcal{A}, \gamma \rrbracket$ can simulate all the evolutions of π .

Proposition 18 (Completeness 1). *Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ and $\pi^1 = (\gamma^1, \delta^1)$. Then, $P \xrightarrow{\tau \backslash \dots \tau \backslash \tau} P' = \llbracket \mathcal{A}, \gamma^1 \rrbracket$.*

Now, we extend the previous result to a series of transitions.

Corollary 19 (Completeness 2). *Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ and $\pi^j = (\gamma^j, \delta^j)$. Then, $P \xrightarrow{\tau \backslash \dots \tau \backslash \tau}^j P'' = \llbracket \mathcal{A}, \gamma^j \rrbracket$.*

5. Examples

Semantically, the topmost restriction $(\nu \text{ names})$ filters out any interaction with virtual links, and releases a private interaction among all participants where all the channel names in the transition labels are hidden (their occurrences are all replaced by τ). This amounts to require that only complete chains are computed by the interaction. In this section, for simplicity, we shall often omit topmost restrictions $(\nu \text{ names})$ from our encoding, but we shall take into account only transitions whose labels are complete chains, i.e. they do not contain virtual links and start/end with the τ action symbol. This way it is possible to observe all channel names that occur in the interaction.

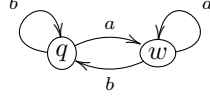


Figure 3: Minimal deterministic labelled transition system.

5.1. Labelled transition system

This example is inspired by the example in [1], where a deterministic transition system is encoded in the Reaction System framework. Here we consider the minimal deterministic transition system in Figure 3.

At the level of RSs, the set of entities to consider is the union of sets of states and of labels of the transition system. Moreover, there is one reaction for each transition: its reactant set consists of the source state and transition label, its inhibitor set includes every other state and label, and its product set is the singleton with the target state. For the transition system in Fig. 3, we take $S = \{q, w, a, b\}$ and the reactions are as follows:

$$\begin{array}{ll} 1 & (\{q, a\}, \{w, b\}, \{w\}) \\ 2 & (\{q, b\}, \{w, a\}, \{q\}) \\ 3 & (\{w, a\}, \{q, b\}, \{w\}) \\ 4 & (\{w, b\}, \{q, a\}, \{q\}) \end{array}$$

Next we show how the above RS is encoded in cCNA.

Encoding of the reactions. The encoding of the reactions is given in a parametric way, with $n \in \{1, 2, 3, 4\}$:

$$\begin{aligned} P_n(q, b, w, a, q) &\triangleq v_n(q, b, w, a, q) \cdot P_n(q, b, w, a, q) \\ &+ \sum_{x \in \{\bar{q}, \bar{b}, w, a\}} v'_n(x) \cdot P_n(q, b, w, a, q) \end{aligned}$$

where

$$\begin{aligned} v_n(q, b, w, a, q) &\triangleq r_n \setminus_{q_i} \square \setminus_{q_o} \square \setminus_{b_i} \square \setminus_{b_o} \square \setminus_{w_i} \square \setminus_{w_o} \square \setminus_{a_i} \square \setminus_{a_o} \square \setminus_{r_{n+1}} \square \setminus_{p_n} \square \setminus_{\bar{q}_i} \square \setminus_{\bar{q}_o} \square \setminus_{p_{n+1}} \square \\ v'_n(x) &\triangleq r_n \setminus_{x_i} \square \setminus_{x_o} \square \setminus_{r_{n+1}} \square \setminus_{p_n} \square \setminus_{p_{n+1}} \square \end{aligned}$$

Then, we have

$$\begin{array}{ll} P_1 &\triangleq P_1(q, a, w, b, w) & P_3 &\triangleq P_3(w, a, q, b, w) \\ P_2 &\triangleq P_2(q, b, w, a, q) & P_4 &\triangleq P_4(w, b, q, a, q) \end{array}$$

and we put, as usual, $r_5 = cxt$ and $p_5 = \tau$.

Encoding of the entities. As for reactions, also the encoding of the entities is given in a parametric way. Here we differentiate the encoding for the entities that are not provided by the context and that can be produced by the reactions, and the ones that can be provided by the context and that are not produced by the reactions.

Here, for the entities q and w that are not provided by the context, we let:

$$\begin{aligned} P_q &\triangleq E(q, \tilde{q}) & \bar{P}_q &\triangleq E(\bar{q}, \tilde{q}) \\ P_w &\triangleq E(w, \tilde{w}) & \bar{P}_w &\triangleq E(\bar{w}, \tilde{w}) \end{aligned}$$

where:

$$\begin{aligned} E(q, \tilde{q}) &\triangleq \sum_{h=1}^3 (q_i \setminus \square_{q_o} \setminus \square)^h \tilde{q}_i \setminus \tilde{q}_o . P_q \\ &+ \sum_{h=1}^3 (q_i \setminus \square_{q_o} \setminus \square)^h . \bar{P}_q \end{aligned}$$

In fact the presence/absence of q and w will be exploited by at least one reaction and at most three reactions.

Here, for the entities a and b that can be provided by the context but not produced by reactions, we let:

$$\begin{aligned} P_a &\triangleq E(a, \hat{a}, \underline{a}) & \bar{P}_a &\triangleq E(\bar{a}, \hat{a}, \underline{a}) \\ P_b &\triangleq E(b, \hat{b}, \underline{b}) & \bar{P}_b &\triangleq E(\bar{b}, \hat{b}, \underline{b}) \end{aligned}$$

where:

$$\begin{aligned} E(a, \hat{a}, \underline{a}) &\triangleq \sum_{h=1}^3 (a_i \setminus \square_{a_o} \setminus \square)^h \hat{a}_i \setminus \hat{a}_o . P_a \\ &+ \sum_{h=1}^3 (a_i \setminus \square_{a_o} \setminus \square)^h \underline{a}_i \setminus \underline{a}_o . \bar{P}_a \end{aligned}$$

Finally, for the context, the encoding follows:

$$Cxt \triangleq cxt \setminus \square_{\hat{a}_i} \setminus \square_{\hat{a}_o} \setminus \square_{\hat{b}_i} \setminus \square_{\hat{b}_o} \setminus p_1 . Cxt \quad + \quad cxt \setminus \square_{\hat{b}_i} \setminus \square_{\hat{b}_o} \setminus \square_{\hat{a}_i} \setminus \square_{\hat{a}_o} \setminus p_1 . Cxt$$

Notice that we exploit here the capabilities of the process algebraic framework to define a nondeterministic, recursive context. We model the context to always offer either a or b , but never both the entities together. The reason is that in the other cases (providing both a and b or neither of them) would lead the system to be stuck because of the simplifications we have adopted in the other processes.

Now, we assume that we have an initial configuration containing the entities q and b :

$$Sys \triangleq I \mid P_q \mid \bar{P}_w \mid \bar{P}_a \mid P_b \mid P_1 \mid P_2 \mid P_3 \mid P_4 \mid Cxt.$$

Then, only the second reaction can be applied, and the transition carries the complete label v below

$$\tau \setminus r_1 \setminus \bar{a}_i \setminus \bar{a}_o \setminus r_2 \setminus q_i \setminus q_o \setminus b_i \setminus b_o \setminus \bar{w}_i \setminus \bar{w}_o \setminus \bar{a}_i \setminus \bar{a}_o \setminus r_3 \setminus \bar{a}_i \setminus \bar{a}_o \setminus r_4 \setminus \bar{w}_i \setminus \bar{w}_o \setminus cxt \setminus \hat{a}_i \setminus \hat{a}_o \setminus \underline{b}_i \setminus \underline{b}_o \setminus p_1 \setminus p_2 \setminus \hat{q}_i \setminus \hat{q}_o \setminus p_3 \setminus p_4 \setminus \tau$$

The parts in bold are provided by the entity processes, the other parts are provided by the processes encoding the reactions and by the process encoding the context (starting at cxt and ending at p_1 . In the label we can read that reactions 1 and 4 have been not executed because the entity a is absent, the reaction 3 has been not applied because the entity w is absent, then only reaction 2 has been applied, and it has produced the entity q . Also, the context provides entity a , that will be available in the next state, and not the entity b . Now, to let the label more readable, we show the result of the application of the function $flat(\cdot)$ to it:

$$r_1 \bar{a} r_2 q b \bar{w} \bar{a} r_3 \bar{a} r_4 \bar{w} cxt \hat{a} \underline{b} p_1 p_2 \tilde{q} p_3 p_4.$$

5.2. A biological toy example of gene expression

We consider a biological toy example in the style of gene's alternative splicing [20]. Alternative splicing is a regulated process during gene expression that results in a single gene coding for multiple proteins. In practice, particular exons of a gene may be included within or excluded from the final processed messenger RNA (mRNA) produced from that gene. In our example, a gene a codes for a protein T when molecules G is present and C is absent, and in the opposite situation a codes for protein T' . This behavior is encoded in reactions 1 and 2. Then, reaction 3 codes for the production of C when proteins T and F are present, and T' absent; reaction 4 codes for the production of G when proteins T' is present and F is absent.

Encoding of the reactions. The encoding of reactions is given in a parametric way:

$$P_n(a, G, C, T) \triangleq \pi_n(a, G, C, T) \cdot P_n(a, G, C, T) + \sum_{x \in \{a, G, C\}} \pi'_n(x) \cdot P_n(a, G, C, T)$$

where

$$\begin{aligned} \pi_n(a, G, C, T) &\triangleq r_n \setminus_{a_i} \setminus_{\square} \setminus_{a_o} \setminus_{G_i} \setminus_{\square} \setminus_{G_o} \setminus_{C_i} \setminus_{\square} \setminus_{C_o} \setminus_{r_{n+1}} \setminus_{\square} \setminus_{p_n} \setminus_{\square} \setminus_{\tilde{T}_i} \setminus_{\square} \setminus_{\tilde{T}_o} \setminus_{p_{n+1}} \\ \pi'_n(x) &\triangleq r_n \setminus_{x_i} \setminus_{\square} \setminus_{x_o} \setminus_{r_{n+1}} \setminus_{\square} \setminus_{p_n} \setminus_{p_{n+1}} \end{aligned}$$

Then we have

$$P_1 \triangleq P_1(a, G, C, T) \quad P_2 \triangleq P_2(a, C, G, T') \quad P_3 \triangleq P_3(F, T, T', C)$$

$$\begin{aligned} P_4 &\triangleq r_4 \setminus_{T'_i} \setminus_{\square} \setminus_{T'_o} \setminus_{\square} \setminus_{F_i} \setminus_{\square} \setminus_{F_o} \setminus_{cxt} \setminus_{\square} \setminus_{p_4} \setminus_{\square} \setminus_{\tilde{G}_i} \setminus_{\square} \setminus_{\tilde{G}_o} \setminus_{\tau} \cdot P_4 \\ &+ r_4 \setminus_{\tilde{T}'_i} \setminus_{\square} \setminus_{\tilde{T}'_o} \setminus_{\square} \setminus_{cxt} \setminus_{\square} \setminus_{p_4} \setminus_{\tau} \cdot P_4 + r_4 \setminus_{F_i} \setminus_{\square} \setminus_{F_o} \setminus_{cxt} \setminus_{\square} \setminus_{p_4} \setminus_{\tau} \cdot P_4 \end{aligned}$$

Encoding of the entities. As for reactions, also the encoding of the entities is given in a parametric way. Here we differentiate three types of encodings: (1) for the entities that are not provided by the context and can be produced by the reactions; (2) for the entities that can be provided by the context and can be produced by the reactions; (3) for the entities that are only provided by the context.

Here, the entities T and T' that can be produced by the reactions and that are not provided by the context:

$$P(T, \tilde{T}) \triangleq \sum_{h=0}^1 (T_i \setminus_{\square} \setminus_{T_o})^h \tilde{T}_i \setminus_{\tilde{T}_o} \cdot P(T, \tilde{T}) + T_i \setminus_{T_o} \cdot P(\bar{T}, \tilde{T})$$

Then, we have

$$P_T \triangleq P(T, \tilde{T}) \quad \bar{P}_T \triangleq P(\bar{T}, \tilde{T}) \quad P_{T'} \triangleq P(T', \tilde{T}') \quad \bar{P}_{T'} \triangleq P(\bar{T}', \tilde{T}')$$

The entities that can be produced by the reactions and that can be provided by the context are as follows:

$$\begin{aligned}
P(C, \widehat{C}, \underline{C}, \widetilde{C}) &\triangleq \sum_{h=0}^1 (C_i \setminus \square_{\widehat{C}_o} \setminus \square)^h \widehat{C}_i \setminus \square_{\widehat{C}_o} \setminus (\widetilde{C}_i \setminus \square_{\widetilde{C}_o} \setminus \square)^h . P(C, \widehat{C}, \underline{C}, \widetilde{C}) \\
&+ \sum_{h=0}^1 (C_i \setminus \square_{\widehat{C}_o} \setminus \square)^h \underline{C}_i \setminus \underline{C}_o . P(\overline{C}, \widehat{C}, \underline{C}, \widetilde{C}) \\
&+ \sum_{h=0}^1 (C_i \setminus \square_{\widehat{C}_o} \setminus \square)^h \underline{C}_i \setminus \underline{C}_o \setminus \square \setminus \widehat{C}_i \setminus \widetilde{C}_o . P(C, \widehat{C}, \underline{C}, \widetilde{C})
\end{aligned}$$

Then, we have

$$\begin{aligned}
P_C &\triangleq P(C, \widehat{C}, \underline{C}, \widetilde{C}) & P_G &\triangleq P(G, \widehat{G}, \underline{G}, \widetilde{G}) \\
\overline{P}_C &\triangleq P(\overline{C}, \widehat{C}, \underline{C}, \widetilde{C}) & \overline{P}_G &\triangleq P(\overline{G}, \widehat{G}, \underline{G}, \widetilde{G})
\end{aligned}$$

The encoding of the entity F that can only be provided by the context follows:

$$\begin{aligned}
P_F &\triangleq \sum_{h=0}^1 (F_i \setminus \square_{\widehat{F}_o} \setminus \square)^h \widehat{F}_i \setminus \widehat{F}_o . P_F + \sum_{h=0}^1 (F_i \setminus \square_{\widehat{F}_o} \setminus \square)^h \underline{F}_i \setminus \underline{F}_o . \overline{P}_F \\
\overline{P}_F &\triangleq \sum_{h=0}^1 (\overline{F}_i \setminus \square_{\widehat{F}_o} \setminus \square)^h \widehat{F}_i \setminus \widehat{F}_o . P_F + \sum_{h=0}^1 (\overline{F}_i \setminus \square_{\widehat{F}_o} \setminus \square)^h \underline{F}_i \setminus \underline{F}_o . \overline{P}_F
\end{aligned}$$

Also in this example we account for a nondeterministic context that can (nondeterministically) provide any combination of the entities C , G F :

$$Cxt \triangleq \sum_{\substack{C^* \in \{C, \overline{C}\} \\ G^* \in \{G, \overline{G}\} \\ F^* \in \{F, \overline{F}\}}} cxt \setminus \square_{C_i^*} \setminus \square_{G_i^*} \setminus \square_{F_i^*} \setminus \square_{p_1}^{F^*} . Cxt$$

Now, to show a possible composition of a transition label, we assume a system where only the entities a , T' , and G are present:

$$Sys \triangleq I \mid P_a \mid \overline{P}_C \mid P_G \mid \overline{P}_F \mid \overline{P}_T \mid P_{T'} \mid P_1 \mid P_2 \mid P_3 \mid P_4 \mid Cxt$$

In the above configuration, reactions 1 and 4 can be applied, and also we assume that the context will provide the entity F , that will be available in the target configuration. Instead of showing the complete transition label, we give its flattened version obtained by applying the function $flat(\cdot)$:

$$r_1 \ a \ G \ \overline{C} \ r_2 \ G \ r_3 \ T' \ r_4 \ T' \ \overline{F} \ cxt \ \underline{C} \ \underline{G} \ \widehat{F} \ p_1 \ \widetilde{T} \ p_2 \ p_3 \ p_4 \ \widetilde{G}.$$

The original label can then be reconstructed just applying the function $unflat(\cdot)$ to the string above.

In [7] we have shown a more complex example, by modeling a RS of a regulatory network for *lac* operon, presented in [3].

6. Bio-simulation

The classical notion of bisimulation for process algebras equates two processes when one process can simulate all the instructions executed by the other one and viceversa. In its weak formulation, internal instructions, i.e. non visible by external observers, are abstracted away. There are many variants of the bisimulation for process algebras, for example the barbed bisimulation [21] only considers the execution of invisible actions, and then equates two processes when they expose the same prefixes; for the mobile ambients [11], a process algebra equipped with a reduction semantics, a notion of behavioural equivalence equates two processes when they expose the same ambients [22].

There are some previous works based on bisimulation applied to models for biological systems. Barbuti et al [23] define a classical setting for bisimulation for two formalisms: the Calculus of Looping Sequences, which is a rewriting system, and the Brane Calculi, which is based on process calculi. Bisimulation is used to verify properties of the regulation of lactose degradation in *Escherichia coli* and the EGF signalling pathway. These calculi allow the authors to model membranes' behaviour. Cardelli et al [24] present two quantitative behavioral equivalences over species of a chemical reaction network with semantics based on ordinary differential equations. Bisimulation identifies a partition where each equivalence class represents the exact sum of the concentrations of the species belonging to that class. Bisimulation also relates species that have identical solutions at all time points when starting from the same initial conditions. Both the fore mentioned formalisms [23, 24] adopt a classical approach to bisimulation. Albeit the bisimulation is a powerful tool for verifying if the behaviour of two different software programs is indistinguishable, in the case of biological systems the classical bisimulation seems to be inappropriate, as the labels of the transitions systems are too concrete. In fact, in a biological soup, a high number of interactions occur at every computational stage, and generally, biologists are only interested to analyse a small subset of them and to focus just on some entities.

For this reason, we propose an alternative notion of bisimulation, that hereafter we call *bio-simulation*, that allows us to compare two biological systems by restricting the observation to only a limited set of events of interest, which can be chosen according to the property one wants to investigate in an experiment. This allows one to tailor the equivalence to different applications and purposes.

The transition labels of our systems record detailed information about all the reactions that have been applied in one transition, about the elements that acted as reagents, as inhibitors or that have been produced, or that have been provided by the context. All these information are stored in the label because they are necessary to compose a transition in a modular way. Depending on the application, only a suitable abstraction over the label can be of interest.

In a way, we want to query our transition labels to extract only the information we care about. To this goal, we introduce a simple language that allows us to formulate detailed and partial queries about what happened in a single transition.

Example 20. For instance we would like to express properties about each step of the bio-simulation of a system like the ones below:

1. Has the entity s_i been used by reaction r_j as reagent?
2. Has the entity s_i been blocked the application of reaction r_j ?
3. Has the entity s_i been produced by reaction r_j ?
4. Has the entity s_i been produced by some reaction?
5. Has the entity s_i been provided by the context?
6. Has the reaction r_j not been applied?

As detailed before, in the following we assume that: (i) the context can be non-deterministic, otherwise it makes little sense to rely on bisimulation to observe the branching structure of system dynamics; (ii) we are interested in observing the names of the entities involved in the transitions and also the reactions that have been applied, thus we assume top level restrictions are absent and rely on solid transitions only (with leftmost and rightmost silent actions).

6.1. Assertion language

Next, we introduce an assertion language that operates on strings and that combines regular expression operators with conjunction and disjunction. We let $names$ be the set of symbolic names used in our assertion language.

Definition 21 (Assertion Language). Atomic assertions ζ and general assertions F are built from the syntax below:

$$\begin{aligned}\zeta &::= \eta \mid ? \mid [N] \\ F &::= \epsilon \mid \zeta \mid F :: F \mid F^+ \mid F^* \mid F \vee F \mid F \wedge F\end{aligned}$$

where $\eta \in names$ and $N \subseteq names$.

Roughly, an atomic assertion ζ denotes either a string composed by a single name η (one of the symbols in the set $names$ for denoting a particular entity, reaction, production or context), or the wildcard $?$ that stands for any symbol, or the pattern $[N]$ that stands for any string composed by a single symbol in the set N . Clearly $?$ is just a shorthand for $[names]$. We write $\mathbf{0}$ for $[\emptyset]$ and $[s_1, \dots, s_n]$ instead of $[\{s_1, \dots, s_n\}]$.

An assertion F is either the empty string ϵ , an atomic assertion ζ , the concatenation of two assertions $F_1 :: F_2$, the replication of F for 1 or more times F^+ , the replication of F for 0 or more times F^* , the disjunction of two assertions $F_1 \vee F_2$ or their conjunction $F_1 \wedge F_2$. We denote by \star the assertion $?$.

An assertion denotes a set of strings over the alphabet $names$ as expected. Below we let $\wp(X)$ denote the powerset of a set X .

Definition 22 (Semantics of Assertions). We define $\llbracket F \rrbracket \subseteq \wp(names^*)$ by induction on the structure of F :

$$\begin{aligned}\llbracket \epsilon \rrbracket &\triangleq \{\epsilon\} & \llbracket F_1 :: F_2 \rrbracket &\triangleq \{\omega_1 :: \omega_2 \mid \omega_1 \in \llbracket F_1 \rrbracket \wedge \omega_2 \in \llbracket F_2 \rrbracket\} \\ \llbracket \alpha \rrbracket &\triangleq \{\alpha\} & \llbracket F^+ \rrbracket &\triangleq \llbracket F \rrbracket^+ \\ \llbracket ? \rrbracket &\triangleq names & \llbracket F^* \rrbracket &\triangleq \llbracket F \rrbracket^* \\ \llbracket [N] \rrbracket &\triangleq N & \llbracket F_1 \vee F_2 \rrbracket &\triangleq \llbracket F_1 \rrbracket \cup \llbracket F_2 \rrbracket \\ & & \llbracket F_1 \wedge F_2 \rrbracket &\triangleq \llbracket F_1 \rrbracket \cap \llbracket F_2 \rrbracket\end{aligned}$$

Definition 23 (Satisfaction as Membership). Let v be a transition label, and F be an assertion. We write $v \models F$ (read as the transition label v satisfies the assertion F) if $flat(v) \in \llbracket F \rrbracket$, otherwise we write $v \not\models F$ (or also $v \models \neg F$) and say that F does not hold at v .

Given two transition labels v, w we write $v \equiv_F w$ if $v \models F \Leftrightarrow w \models F$, i.e. if both v, w satisfy F or they do not.

Example 24. The assertions corresponding to the sample queries listed in Example 20 are as follows:

1. $\star :: r_j :: [s_1, \dots, s_n]^* :: s_i :: [s_1, \dots, s_n]^* :: [\bar{s}_1, \dots, \bar{s}_n]^+ :: \star$
2. $\star :: r_j :: [s_i, \bar{s}_i] :: r_{j+1} :: \star$
3. $\star :: p_j :: [ents]^* :: \tilde{s}_i :: \star$
4. $\star :: \tilde{s}_i :: \star$
5. $\star :: \hat{s}_i :: \star$
6. $\star :: r_j :: ? :: r_{j+1} :: \star$

where in 1, 2, 6 we exploit the fact that in a reaction (R, I, P) the sets R of reactants and I of inhibitors are non empty, so that if there is only one symbol between the occurrence of r_j and r_{j+1} it means the reaction r_j has not been applied. Viceversa, if the reaction r_j has been applied the occurrence of r_j must be followed by at least one of the symbols in $\{s_1, \dots, s_n\}$ and then by at least one of the symbols in $\{\bar{s}_1, \dots, \bar{s}_n\}$.

6.2. Bio-similarity and bio-logical equivalence

The notion of bio-simulation builds on the above language of assertions to parameterize the induced equivalence on the property of interest. Please recall that we have defined the behaviour of the context in a non deterministic way, thus at each step, different possible sets of entities can be provided to the system and different sets of reactions can be enabled/disabled. Bio-simulation can thus be used to compare the behaviour of different systems that share some of the reactions or entities or also to compare the behaviour of the same set of reactions when different contexts are provided.

Definition 25 (Bio-similarity \sim_F). Given an assertion F , a *bio-simulation* \mathbf{R}_F that respects F is a binary relation over cCNA processes such that, whenever $P \mathbf{R}_F Q$ then:

- for any v, P' such that $P \xrightarrow{v} P'$ then there exist w, Q' such that $Q \xrightarrow{w} Q'$ with $v \equiv_F w$ and $P' \mathbf{R}_F Q'$.
- for any w, Q' such that $Q \xrightarrow{w} Q'$ then there exist v, P' such that $P \xrightarrow{v} P'$ with $v \equiv_F w$ and $P' \mathbf{R}_F Q'$.

We let \sim_F denote the largest bio-simulation and we say that P is *bio-similar* to Q , with respect to F , if $P \sim_F Q$.

Remark 26. Please remember that the notation $P \xRightarrow{v} P'$ refers to ordinary transitions $P \xrightarrow{v} P'$ where v is a complete chain (solid and with τ actions at the extremes). The double arrow notation should not be confused with the notation for weak transitions commonly found in the literature on process algebras.

Remark 27. An alternative way to look at a bio-simulation that respects F is to define it as an ordinary bisimulation over the transition system labelled over $\{F, \neg F\}$ obtained by transforming each transition $P \xRightarrow{v} P'$ such that $v \models F$ into $P \xrightarrow{F} P'$ and each transition $P \xRightarrow{v} P'$ such that $v \not\models F$ into $P \xrightarrow{\neg F} P'$.

It can be easily shown that the identity relation is a bio-simulation and that bio-simulations are closed under (relational) inverse, composition and union and that, as a consequence, bio-similarity is an equivalence relation.

Now, we introduce a slightly modified version of the Hennessy Milner Logic (HML) [15], called bioHML; due to the reasons we explained above, we do not want to look at the complete transition labels, thus we rely on our simple assertion language to make it parametric w.r.t the assertion F of interest:

Definition 28 (BioHML). Let F be an assertion, then the set of bioHML formulas G that respects F are built by the following syntax:

$$\begin{aligned} \chi &::= F \mid \neg F \\ G, H &::= t \mid f \mid G \wedge H \mid G \vee H \mid \langle \chi \rangle G \mid [\chi] G \end{aligned}$$

Remark 29. An alternative way to look at bioHML formulas is as ordinary HML formulas over the set of labels $\{F, \neg F\}$.

As usual, the semantics of a bioHML formula is the set of cCNA processes that satisfy it.

Definition 30 (Semantics of BioHML). We define $\llbracket G \rrbracket \subseteq \mathcal{P}$ by induction on the structure of G :

$$\begin{aligned} \llbracket t \rrbracket &\triangleq \mathcal{P} & \llbracket G \wedge H \rrbracket &\triangleq \llbracket G \rrbracket \cap \llbracket H \rrbracket \\ \llbracket f \rrbracket &\triangleq \emptyset & \llbracket G \vee H \rrbracket &\triangleq \llbracket G \rrbracket \cup \llbracket H \rrbracket \end{aligned}$$

$$\begin{aligned} \llbracket \langle \chi \rangle G \rrbracket &\triangleq \{P \in \mathcal{P} : \exists v, P'. P \xRightarrow{v} P' \text{ with } v \models \chi \text{ and } P' \in \llbracket G \rrbracket\} \\ \llbracket [\chi] G \rrbracket &\triangleq \{P \in \mathcal{P} : \forall v, P'. P \xRightarrow{v} P' \text{ implies } v \models \chi \text{ and } P' \in \llbracket G \rrbracket\} \end{aligned}$$

We write $P \models G$ (P satisfies G) if and only if $P \in \llbracket G \rrbracket$.

Negation is not included in the syntax, but the converse \overline{G} of a bioHML formula G can be easily defined inductively in the same way as for HML logic.

Definition 31 (Converse). Given a bioHML formula G we define its converse \overline{G} as follows:

$$\begin{aligned} \overline{t} &\triangleq f & \overline{G \wedge H} &\triangleq \overline{G} \vee \overline{H} & \overline{\langle \chi \rangle G} &\triangleq [\chi] \overline{G} \\ \overline{f} &\triangleq t & \overline{G \vee H} &\triangleq \overline{G} \wedge \overline{H} & \overline{[\chi] G} &\triangleq \langle \chi \rangle \overline{G} \end{aligned}$$

We observe that, as expected, for any bioHML formula G and process P we have $\overline{\overline{G}} = G$ and $P \models \overline{\overline{G}}$ iff $P \not\models G$.

Definition 32 (Bio-logical equivalence). We let \mathcal{L}_F be the set of all bioHML formulas that respects F and we say that two processes P, Q are *bio-logically equivalent w.r.t. F*, written $P \equiv_{\mathcal{L}_F} Q$, when P and Q satisfy exactly the same bioHML formulas in \mathcal{L}_F , i.e. when for any $G \in \mathcal{L}_F$ we have $P \models G \Leftrightarrow Q \models G$.

Finally, we extend the classical result establishing the correspondence between the logical equivalence induced by HML with bisimilarity for proving that bio-similarity coincides with bio-logical equivalence.

Theorem 33 (Correspondence). $\sim_F = \equiv_{\mathcal{L}_F}$

6.3. Bio-simulation at work

We will show how bio-simulation works. For the sake of space, we consider a very simple example with only two reactions. Reaction P_1 requires G to produce C ; reaction P_2 requires C to produce G ; both reactions have H as inhibitor. Now, we set two systems defined by the same two reactions, with the two different initial configuration, and with two different context definitions. The two reactions work as follows (where we omit to specify the cases where H is present, as they will never happen):

$$\begin{aligned} P_1 &\triangleq \tau \backslash \frac{\square}{G_i} \backslash \frac{G_o}{\square} \backslash \frac{\bar{H}_o}{\bar{H}_i} \backslash \frac{\square}{\square} \backslash \frac{p_1}{r_2} \backslash \frac{\square}{\bar{C}_i} \backslash \frac{\bar{C}_o}{\square} \backslash p_2 . P_1 \quad + \quad \tau \backslash \frac{\square}{\bar{G}_i} \backslash \frac{\bar{G}_o}{\square} \backslash \frac{p_1}{r_2} \backslash \frac{\square}{\square} \backslash p_2 . P_1 \\ P_2 &\triangleq r_2 \backslash \frac{\square}{\bar{C}_i} \backslash \frac{C_o}{\square} \backslash \frac{\bar{H}_o}{\bar{H}_i} \backslash \frac{\square}{\square} \backslash \frac{\square}{cxt} \backslash \frac{p_2}{\square} \backslash \frac{\bar{C}_o}{\bar{G}_i} \backslash \frac{\bar{C}_o}{\square} \backslash \tau . P_2 \quad + \quad r_2 \backslash \frac{\square}{\bar{C}_i} \backslash \frac{\bar{C}_o}{\square} \backslash \frac{\square}{cxt} \backslash \frac{p_2}{\square} \backslash \tau . P_2 \end{aligned}$$

The two contexts follow :

$$\begin{aligned} Cxt &\triangleq \text{cxt} \backslash \frac{\square}{\widehat{C}_i} \backslash \frac{\square}{\widehat{C}_o} \backslash \frac{\square}{\underline{G}_i} \backslash \frac{\square}{\underline{G}_o} \backslash \frac{\square}{\underline{H}_i} \backslash \frac{\square}{\underline{H}_o} \backslash_{p_1}. Cxt + \text{cxt} \backslash \frac{\square}{\underline{C}_i} \backslash \frac{\square}{\underline{C}_o} \backslash \frac{\square}{\underline{G}_i} \backslash \frac{\square}{\underline{G}_o} \backslash \frac{\square}{\underline{H}_i} \backslash \frac{\square}{\underline{H}_o} \backslash_{p_1}. Cxt \\ Cxt' &\triangleq \text{cxt} \backslash \frac{\square}{\widehat{G}_i} \backslash \frac{\square}{\widehat{G}_o} \backslash \frac{\square}{\underline{C}_i} \backslash \frac{\square}{\underline{C}_o} \backslash \frac{\square}{\underline{H}_i} \backslash \frac{\square}{\underline{H}_o} \backslash_{p_1}. Cxt' + \text{cxt} \backslash \frac{\square}{\underline{G}_i} \backslash \frac{\square}{\underline{G}_o} \backslash \frac{\square}{\underline{C}_i} \backslash \frac{\square}{\underline{C}_o} \backslash \frac{\square}{\underline{H}_i} \backslash \frac{\square}{\underline{H}_o} \backslash_{p_1}. Cxt' \end{aligned}$$

The definition of the processes encoding G and C is similar, and it is given in a parametric way:

$$P(G) \triangleq G_i \setminus_{G_o} \bar{P}(G) \qquad \bar{P}(G) \triangleq \bar{G}_i \setminus_{\bar{G}_o} \tilde{G}_i \setminus_{\tilde{G}_o} P(G)$$

and we have $P_G \triangleq P(G)$, $P_C \triangleq P(C)$, then $\overline{P}_H \triangleq \overline{H}_i \setminus_{\overline{H}_o} \cdot \overline{P}_H + \overline{H}_i \setminus_{\overline{H}_o} \square \setminus_{\overline{H}_i} \cdot \overline{P}_H$, as H is neither produced nor provided by the context. Then, the initial configuration of system Sys_1 includes G and not C and the context can only provide C , the initial configuration of system Sys_2 includes C and not G and the context can only provide G :

$$\begin{array}{lcl} Sys_1 & \triangleq & I \mid P_1 \mid P_2 \mid P_G \mid \overline{P}_C \mid \overline{P}_H \mid Cxt \\ Sys_2 & \triangleq & I \mid P_1 \mid P_2 \mid \overline{P}_G \mid P_C \mid \overline{P}_H \mid Cxt' \end{array}$$

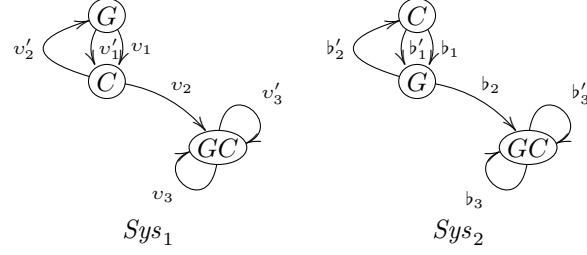


Figure 4: The operational semantics of Sys_1 and Sys_2 .

In Figure 4 we show the operational semantics of Sys_1 (on the left) and Sys_2 (on the right), limited to the transitions with complete and solid labels. To improve readability, we named the states of the transition system with the entities that are present in the state. For example, in the leftmost figure, since in Sys_1 only the entity G is available, we name the topmost state G instead of Sys_1 . Similarly, for the rightmost figure, where we write, e.g. C instead of Sys_2 . As before we show the output of the $flat(\cdot)$ function applied to the transition labels:

$$\begin{aligned}
 flat(v_1) &\triangleq r_1 \ G \ \overline{H} \ r_2 \ \overline{C} \ \text{ctx} \ \widehat{C} \ \underline{G} \ \underline{H} \ p_1 \ \widetilde{C} \ p_2 \\
 flat(v'_1) &\triangleq r_1 \ G \ \overline{H} \ r_2 \ \overline{C} \ \text{ctx} \ \underline{C} \ \underline{G} \ \underline{H} \ p_1 \ \widetilde{C} \ p_2 \\
 flat(v_2) &\triangleq r_1 \ \overline{G} \ r_2 \ C \ \overline{H} \ \text{ctx} \ \widehat{C} \ \underline{G} \ \underline{H} \ p_1 \ p_2 \ \widetilde{G} \\
 flat(v'_2) &\triangleq r_1 \ \overline{G} \ r_2 \ C \ \overline{H} \ \text{ctx} \ \underline{C} \ \underline{G} \ \underline{H} \ p_1 \ p_2 \ \widetilde{G} \\
 flat(v_3) &\triangleq r_1 \ G \ \overline{H} \ r_2 \ C \ \overline{H} \ \text{ctx} \ \widehat{C} \ \underline{G} \ \overline{H} \ p_1 \ \widetilde{C} \ p_2 \ \widetilde{G} \\
 flat(v'_3) &\triangleq r_1 \ G \ \overline{H} \ r_2 \ C \ \overline{H} \ \text{ctx} \ \underline{C} \ \underline{G} \ \overline{H} \ p_1 \ \widetilde{C} \ p_2 \ \widetilde{G}
 \end{aligned}$$

The labels b_i, b'_i , with $i \in \{1, 2, 3\}$ can be obtained by labels v_i, v'_i by substituting G with C and viceversa.

Now, it is easy to check that Sys_1 and Sys_2 are bio-similar w.r.t the property F saying that G and C are simultaneously produced, formally: $Sys_1 \sim_F Sys_2$ with $F = \star :: \widetilde{G} :: \star \wedge \star :: \widetilde{C} :: \star$.

On the contrary, $Sys_1 \not\sim_{F'} Sys_2$ with $F' = \star :: \widetilde{C} :: \star$, because it happens that both transition labels ℓ_1 and ℓ'_1 , in Sys_1 , record the production of C , whereas transition labels b_1 and b'_1 do not. In fact, the bioHML formula $G \triangleq \langle F' \rangle \mathbf{t}$ can be used to distinguish Sys_1 from Sys_2 , as $Sys_1 \models G$ and $Sys_2 \not\models G$.

7. Towards Enhanced Reaction Systems

Our encoding increases the expressivity of RS concerning: the possibility of alternative behaviour of mutated entities, and the communication between two different RSs. It is important to note that, when the context is deterministic, our encoding guarantees that from each state, in the cCNA transition system, only one state is reachable, as the dynamics is totally deterministic.

7.1. Mutating entities

In RS, when an entity is present, it can potentially be involved in each reactions where it is required. With a few more lines of code, in cCNA it is possible to describe the behaviour of a mutation of an entity, in a way that the mutated version of the entity can take part to only a subset of the reactions requiring the *normal version* of the entity. For example, let us assume that entity $s1$ is consumed by reactions $a1$ and $a2$. Reaction $a1$ produces also $s1$ if $s2$ is present, otherwise $a1$ produces a mutated version of $s1$, say $s1'$. When $s1'$ is produced, reaction $a2$ behaves in the same way as if $s1$ would be absent, whereas $a2$ recognises the presence of $s1'$ and behaves in the same way as if $s1$ would be present. Technically, in both cases it is enough to add one more nondeterministic choice in the code of P_{a1} and P_{a2} .

7.2. Communicating Reaction Systems

We sketch how it is possible to program two RSs encodings, in a way that the entities that usually come from the context of one RS will be provided instead from the other RS.

Example 34. Let $rs1$ and $rs2$ be two RSs, defined, respectively, by the reactions $a_1 = (s, \emptyset, x)$ and $a_2 = (y, \emptyset, s)$. Now, we set our example such that the two contexts, for $rs1$ and $rs2$, do not provide any entities. We also assume that entity s in $rs1$ is provided by $rs2$, as $rs2$ produces a quantity of s that is enough for $rs1$ and $rs2$. For technical reasons, we can not use the same name for s in both the two RSs, then we use the name ss in $rs2$. We need to modify our encoding technique to suit this new setting. As we do not model contexts, we introduce *dummy* channel names dx and dss to model the absence of entities. Also, thanks to the simplicity of the example, we can leave out the use of the p_i channels. This streamlining does not affect the programming technique we propose to make two RSs communicate. First, we translate the reaction in $rs1$, by setting $\llbracket a_1 \rrbracket \triangleq P_{a_1}$ where:

$$P_{a_1} \triangleq \tau \backslash s_i \backslash \square \backslash s_o \backslash \square \backslash \tilde{x}_o \backslash \square \backslash a_2 \cdot P_{a_1} + \tau \backslash \bar{s}_i \backslash \square \backslash \tilde{s}_o \backslash \square \backslash dx_o \backslash \square \backslash a_2 \cdot P_{a_1}$$

Please note, that prefixes of process P_{a_1} end with the channel name a_2 , as the link chain is now connected with the reaction of $rs2$. The encoding for the entities is given by setting $\llbracket s \rrbracket \triangleq P_s$ and $\llbracket x \rrbracket \triangleq P_x$, where:

$$\begin{aligned} P_s &\triangleq s_i \backslash \square \backslash s_o \backslash \square \backslash \tilde{s}_i \backslash \square \backslash \tilde{s}_o \cdot P_s + s_i \backslash s_o \cdot \overline{P_s} \\ \overline{P_s} &\triangleq \bar{s}_i \backslash \square \backslash \tilde{s}_o \backslash \square \backslash \tilde{s}_i \backslash \square \backslash \tilde{s}_o \cdot P_s + \bar{s}_i \backslash \tilde{s}_i \cdot \overline{P_s} \\ P_x &\triangleq \tilde{x}_i \backslash \tilde{x}_o \cdot P_x + dx_i \backslash dx_o \cdot \overline{P_x} \\ \overline{P_x} &\triangleq \tilde{x}_i \backslash \tilde{x}_o \cdot P_x + dx_i \backslash dx_o \cdot \overline{P_x} \end{aligned}$$

The encoding for $rs2$ is given by $\llbracket a_2 \rrbracket \triangleq P_{a_2}$, where:

$$P_{a_2} \triangleq a_2 \backslash y_i \backslash \square \backslash y_o \backslash \square \backslash \tilde{s}_i \backslash \square \backslash \tilde{s}_o \backslash \tau \cdot P_{a_2} + a_2 \backslash \square \backslash \tilde{y}_o \backslash \square \backslash dss_i \backslash \square \backslash \tau \cdot P_{a_2}$$

In the encoding of the entities in $rs2$, we introduce the mechanism that allows the entity s (ss in $rs2$) to be provided in $rs1$. Every time ss is produced in $rs2$, a virtual link is created to synchronise with $rs1$ on link $\hat{s}_i \setminus \hat{s}_o$. To this purpose we define $\llbracket ss \rrbracket \triangleq P_{ss}$ and $\llbracket y \rrbracket \triangleq P_y$, where:

$$\begin{aligned} \frac{P_{ss}}{P_{ss}} &\triangleq \frac{\tilde{s}s_i \setminus \square \setminus \hat{s}_o \setminus \square \setminus \tilde{s}s_o . P_{ss} + dss_i \setminus dss_o . \overline{P_{ss}}}{\tilde{s}s_i \setminus \hat{s}_i \setminus \hat{s}_o \setminus \tilde{s}s_o . P_{ss} + dss_i \setminus dss_o . \overline{P_{ss}}} \\ \frac{P_y}{P_y} &\triangleq \frac{y_i \setminus y_o . P_y}{\bar{y}_i \setminus \bar{y}_o . P_y} \end{aligned}$$

We now assume that the initial system is $S \triangleq (\nu names)(P_{a_1} | P_{a_2} | P_s | P_y | \overline{P_x} | \overline{P_{ss}})$, i.e. only entities s and y are present. Now, the only possible transition has the following label (that we report without restriction):

$$\tau \setminus \frac{s_i \setminus s_o \setminus \tilde{x}_i \setminus \tilde{x}_o \setminus a_2 \setminus y_i \setminus y_o \setminus \tilde{s}s_i \setminus \hat{s}_i \setminus \hat{s}_o \setminus \tilde{s}s_o}{s_i \setminus s_o \setminus \tilde{x}_i \setminus \tilde{x}_o \setminus a_2 \setminus y_i \setminus y_o \setminus \tilde{s}s_i \setminus \hat{s}_i \setminus \hat{s}_o \setminus \tilde{s}s_o} \tau,$$

where the black links belong to the prefixes of P_{a_1} , and P_{a_2} , the blue links belong to P_s , the gray links belong to P_y , and $\overline{P_x}$ and the red links belong to $\overline{P_{ss}}$. After the execution, the entity s is still present in $rs1$ as it has been provided by $rs2$.

As we have briefly sketched, our model of two *communicating Reaction Systems* can enable the study of the behaviour of one RS in relation to another one. Thus, the products of the reactions of one RS can become the input for another one. This could allow for a modular approach to modeling complex systems, by composing different Reaction Systems.

8. Conclusion

In this paper we have introduced cCNA, that generalises CNA by allowing the use of prefixes that are link chains and not just single links. This extension was initially described in the future work section of [8]. Thanks to this enhancement, cCNA allowed us to define a faithful encoding of Reaction Systems in a process algebraic framework. This encoding shows several benefits. First, contexts of RSs can be easily defined recursively and exhibit non deterministic behaviour. Second, the operational semantics is defined in a compositional way by a set of SOS inference rules. Third, we have defined a new assertion language, which allows us to specify the properties to be verified over the labels of the operational semantics. Assertions can be used to tailor the classical notion of bisimilarity and Hennessy-Milner logic to focus on some particular aspects or experiments. We have called *bio-similarity* the induced notion of equivalence.

We are currently investigating how to integrate our methodology with other formal techniques to prove properties of the modeled systems, along the lines in [25, 26, 27]. Moreover, we are considering possible enhancements of RSs based on entity mutation and on the possibility for two RSs to exchange entities.

As future work, we plan to implement a prototype of our embedding, with an automatic translation from RSs to `link`-calculus, so to exploit the implementation of the symbolic semantics of the `link`-calculus [28] that can be found in [29].

References

- [1] R. Brijder, A. Ehrenfeucht, M. Main, G. Rozenberg, A tour of reaction systems, *International Journal of Foundations of Computer Science* 22 (07) (2011) 1499–1517.
- [2] S. Azimi, B. Iancu, I. Petre, Reaction system models for the heat shock response, *Fundamenta Informaticae* 131 (3-4) (2014) 299–312.
- [3] L. Corolli, C. Maj, F. Marini, D. Besozzi, G. Mauri, An excursion in reaction systems: From computer science to biology, *Theoretical Computer Science* 454 (2012) 95–108.
- [4] S. Azimi, Steady states of constrained reaction systems, *Theor. Comput. Sci.* 701 (C) (2017) 20–26.
- [5] R. Barbuti, R. Gori, F. Levi, P. Milazzo, Investigating dynamic causalities in reaction systems, *Theor. Comput. Sci.* 623 (2016) 114–145.
- [6] F. Okubo, T. Yokomori, The computational capability of chemical reaction automata, *Natural Computing* 15 (2) (2016) 215–224.
- [7] L. Brodo, R. Bruni, M. Falaschi, Enhancing reaction systems: A process algebraic approach, in: M. Alvim, K. Chatzikokolakis, C. Olarte, F. Valencia (Eds.), *The Art of Modelling Computational Systems: A Journey from Logic and Concurrency to Security and Privacy*, Vol. 11760 of *Lecture Notes in Computer Science*, Springer Berlin, 2019, pp. 68–85.
- [8] C. Bodei, L. Brodo, R. Bruni, A formal approach to open multiparty interactions, *Theoretical Computer Science* 763 (2019) 38–65.
- [9] C. Bodei, L. Brodo, R. Bruni, Open multiparty interaction, in: *Recent Trends in Algebraic Development Techniques, 21st International Workshop, WADT 2012*, Vol. 7841 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 1–23.
- [10] C. Bodei, L. Brodo, R. Bruni, The link-calculus for open multiparty interactions, *Information and Computation* (2020) 104587.
- [11] L. Cardelli, A. D. Gordon, Mobile ambients, *Theoretical Computer Science* 240 (1) (2000) 177–213.
- [12] G. Ciobanu, V. A. Zakharov, Encoding mobile ambients into the π -calculus, in: *Perspectives of Systems Informatics*, Vol. 4378, Springer Berlin Heidelberg, 2007, pp. 148–165.

- [13] L. Brodo, On the expressiveness of pi-calculus for encoding mobile ambients, *Mathematical Structures in Computer Science* 28 (2) (2018) 202–240.
- 700 [14] C. Bodei, L. Brodo, R. Bruni, D. Chiarugi, A flat process calculus for nested membrane interactions, *Sci. Ann. Comp. Sci.* 24 (1) (2014) 91–136.
- [15] M. Hennessy, R. Milner, On observing nondeterminism and concurrency, in: J. de Bakker, J. van Leeuwen (Eds.), *Automata, Languages and Programming*, Vol. 85 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1980, pp. 299–309.
- 705 [16] A. Bernini, L. Brodo, P. Degano, M. Falaschi, D. Hermith, Process calculi for biological processes, *Natural Computing* 17 (2) (2018) 345–373.
- [17] B. Aman, G. Ciobanu, Controlled reversibility in reaction systems, in: M. Gheorghe, G. Rozenberg, A. Salomaa, Z. Claudio (Eds.), *Membrane Computing*, Springer International Publishing, 2018, pp. 40–53.
- 710 [18] P. Bottoni, A. Labella, G. Rozenberg, Networks of reaction systems, *International Journal of Foundations of Computer Science* 31 (01) (2020) 53–71.
- [19] A. Męski, W. Penczek, G. Rozenberg, Model checking temporal properties of reaction systems, *Information Sciences* 313 (2015) 22–42.
- 715 [20] J. D. Watson, T. A. Baker, S. P. Bell, *Molecular Biology of the Gene*, Pearson Education, USA, 2013.
- [21] R. Milner, D. Sangiorgi, Barbed bisimulation, in: W. Kuich (Ed.), *Automata, Languages and Programming*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1992, pp. 685–695.
- 720 [22] A. Gordon, L. Cardelli, Equational properties of mobile ambients, *Mathematical Structures in Computer Science* 13 (3) (2003) 371–408.
- [23] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, A. Troina, Bisimulations in calculi modelling membranes, *Formal Aspects of Computing* 20 (4) (2008) 351–377.
- 725 [24] L. Cardelli, M. Tribastone, M. Tschaikowski, A. Vandin, Forward and backward bisimulations for chemical reaction networks, in: *26th International Conference on Concurrency Theory, CONCUR 2015*, Vol. 42, Schloss Dagstuhl- Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, 2015, pp. 226–239.
- 730 [25] D. Chiarugi, M. Falaschi, D. Hermith, C. Olarte, L. Torella, Modelling non-markovian dynamics in biochemical reactions, *BMC Systems Biology* 9 (S-3) (2015) S8.
- [26] C. Olarte, D. Chiarugi, M. Falaschi, D. Hermith, A proof theoretic view of spatial and temporal dependencies in biochemical systems, *Theor. Comput. Sci.* 641 (2016) 25–42.
- 735

- 1
2
3
4
5
6
7
8
9 [27] C. Bodei, L. Brodo, R. Gori, F. Levi, A. Bernini, D. Hermith, A static
10 analysis for Brane Calculi providing global occurrence counting informa-
11 tion, Theoretical Computer Science 696 (2017) 11–51.
12
13 [28] L. Brodo, C. Olarte, Symbolic semantics for multiparty interactions in the
14 link-calculus, in: Proc. of SOFSEM'17, Vol. 10139 of Lecture Notes in Com-
15 puter Science, Springer, 2017, pp. 62–75.
16
17 [29] C. Olarte, SiLVer: Symbolic links verifier, [http://subsell.logic.at/](http://subsell.logic.at/links/links-web/index.html)
18 [links/links-web/index.html](http://subsell.logic.at/links/links-web/index.html) (Dec. 2018).
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

Appendix A. Omitted Proofs

In this section we report the proofs for the results in Section 4 and in Section 6.

Lemma 12. *Let $\mathcal{A} = (S, A)$ be a RS and let $\pi = (\gamma, \delta)$ be an extended interactive process in \mathcal{A} . Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ its cCNA encoding. If exists P' such that $P \xrightarrow{(\nu \text{ names})v} P'$ is a transition of P , then*

1. for each reaction $aj \in A$, the corresponding channels r_j and p_j appear in v ; for each entity $s \in S$, the corresponding channel s (suitably decorated) appear in v ; the channel cxt appears in v ;
2. for each reaction $aj \in A$ and each virtual link offered by processes P_a and Cxt_1 is overlapped by exactly one solid link offered by processes representing entities.

PROOF. We prove the two items separately:

1. by Definition 11, all the names that appear in the prefixes of any subprocess are in the set *names* and thus restricted. They include all the reaction names r_j , all the production names p_j , all the entity names s_i, s_o , in all their decorated versions, and the name cxt . Therefore the chain v must start and end with a τ action and cannot contain virtual links. The only prefix that starts with τ is the one of the recursive init process I (prefix $\tau \setminus_{r_1}$) and the only prefixes that end with τ are those associated to reaction a_u (as we have assumed that $p_{u+1} = \tau$, where u is the number of reactions). Then each prefix that starts with r_j involves p_j and r_{j+1} . Thus all the prefixes associated with reactions must be concatenated and also the prefix associated with the context (remember that $r_{u+1} = cxt$), forming the backbone of the label. Since the context processes is involved, then all entities processes are also involved. Then, all the processes I, P_a, P_s (or \overline{P}_s), and Cxt_1 must participate to each transition.
2. for each reaction $aj \in A$, the cCNA code of P_{aj} leaves one virtual link between two solid links of the types $r_j \setminus \dots \setminus_{s_i} \square \setminus_{s_o} \dots \setminus_{r_{j+1}} \dots \setminus_{p_j} \setminus \dots \setminus_{p_{j+1}}$. Then, it derives that the process P_s , encoding the behaviour of entity s , can participate by filling the virtual link in the above transition by only offering one solid link of the form $s_i \setminus_{s_o}$. In fact, there is no other way to generate a solid chain from s_i to s_o . The same reasoning holds for the processes Cxt_1 and for all the decorated versions of s_i, s_o .

Proposition 16 (Correctness 1). *Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ with*

$$P = (\nu \text{ names}) \left(I \mid \prod_{a \in A} P_a \mid Cxt_1 \mid \prod_{s \in C_0} P_s \mid \prod_{s \notin C_0} \overline{P}_s \right).$$

If there exists P' such that $P \xrightarrow{v} P'$, it holds that:

1. $v = \tau \setminus \dots \tau \setminus \tau$, and
2. $P' = (\nu \text{ names}) (I \mid \prod_{a \in A} P_a \mid Cxt_2 \mid \prod_{s \in C_1 \cup D_1} P_s \mid \prod_{s \notin C_1 \cup D_1} \bar{P}_s)$.

Moreover, given $\pi^1 = (\gamma^1, \delta^1)$, we have $P' = \llbracket \mathcal{A}, \gamma^1 \rrbracket$.

PROOF. First, we note that all the channels in the system are restricted, see Definition 11, then it holds that the transition labels are of the form $v = \tau \setminus \dots \tau \setminus \tau$. Now, by Definition 11 and by Lemma 12.1, all the channels r_j, p_j , with $j \in [1, \dots, u]$, and cxt , and all the annotated versions of s_i, s_o are restricted. Also, processes Cxt_1 always requires the interaction with P_s on either on channels \hat{s}_i, \hat{s}_o or on channels $\underline{s}_i, \underline{s}_o$. It derives that all the processes: P_a (coding the behaviour of reaction $a \in A$), P_s (coding the behaviour of entity $s \in S$), and Cxt_1 (coding the behaviour of the context regarding all the entities) have been involved in the transition.

For any process P_{aj} encoding a reaction aj we have the following cases:

- (a) if aj is applicable and it produces the entity s , the process P_{aj} provides a code of this type:

$$P_{aj} \triangleq r_j \setminus \dots \setminus \square_{r_{j+1}} \setminus \square_{p_j} \setminus \dots \setminus \square_{\tilde{s}_i} \setminus \square_{\tilde{s}_o} \setminus \dots \setminus p_{j+1} \cdot P_{aj};$$

- (b) if aj is applicable and it consumes the entity s , the process P_{aj} provides a code of this type:

$$P_{aj} \triangleq r_j \setminus \dots \setminus \square_{s_i} \setminus \square_{s_o} \setminus \dots \setminus \square_{r_{j+1}} \setminus \square_{p_j} \setminus \dots \setminus p_{j+1} \cdot P_{aj};$$

- (c) if aj is applicable and it requires the absence of the entity s , the process P_{aj} provides a code of this type:

$$P_{aj} \triangleq r_j \setminus \dots \setminus \square_{\bar{s}_i} \setminus \square_{\bar{s}_o} \setminus \dots \setminus \square_{r_{j+1}} \setminus \square_{p_j} \setminus \dots \setminus p_{j+1} \cdot P_{aj};$$

- (d) if aj is not applicable, the process P_{aj} executes a code capturing either the absence of one of its reactants (case 1), or the presence of one of its inhibitors (case 2):

1. $P_{aj} \triangleq r_j \setminus \dots \setminus \square_{\bar{s}_i} \setminus \square_{\bar{s}_o} \setminus \dots \setminus \square_{r_{j+1}} \setminus \square_{p_j} \setminus \dots \setminus p_{j+1} \cdot P_{aj};$
2. $P_{aj} \triangleq r_j \setminus \dots \setminus \square_{s_i} \setminus \square_{s_o} \setminus \dots \setminus \square_{r_{j+1}} \setminus \square_{p_j} \setminus \dots \setminus p_{j+1} \cdot P_{aj}.$

Now, we consider the structure of the process Cxt_1 . By Definition 11, Cxt_1 is the unique process encoding the behaviour of the context regulating the supply of any entity.

- (e) The code of the process Cxt_1 that provides s and not e has the following structure:

$$Cxt_1 \triangleq cxt \setminus \dots \setminus \square_{\hat{s}_i} \setminus \square_{\hat{s}_o} \setminus \dots \setminus \square_{\underline{e}_i} \setminus \square_{\underline{e}_o} \setminus \dots \setminus p_1 \cdot Cxt_2.$$

The code executed by P_s has the following structure:

- (f) $P_s \triangleq \sum_{h,k \geq 0} (s_i \setminus \square_{s_o} \setminus \square)^h \widehat{s}_i \setminus \square_{s_o} \setminus \square (\widetilde{s}_i \setminus \square_{s_o} \setminus \square)^k . P_s$, if $s \in C_{i+1}$;
 (g) $P_s \triangleq \sum_{h \geq 0, k \geq 1} (s_i \setminus \square_{s_o} \setminus \square)^h \underline{s}_i \setminus \square_{s_o} \setminus \square (\widetilde{s}_i \setminus \square_{s_o} \setminus \square)^k . P_s$, if $s \notin C_{i+1}$
 (h) $P_s \triangleq \sum_{h \geq 0} (s_i \setminus \square_{s_o} \setminus \square)^h \underline{s}_i \setminus \square_{s_o} . \overline{P}_s$, if $s \notin C_{i+1}$

where, by Lemma 12.2, h is the number of reactions requiring the presence of s plus possibly some reactions not requiring s ; and k is the number of reactions producing s .

Similarly, the code executed by \overline{P}_s has the following structure:

- (f') $\overline{P}_s \triangleq \sum_{h,k \geq 0} (\widetilde{s}_i \setminus \square_{s_o} \setminus \square)^h \widehat{s}_i \setminus \square_{s_o} \setminus \square (\widetilde{s}_i \setminus \square_{s_o} \setminus \square)^k . P_s$, if $s \in C_{i+1}$;
 (g') $\overline{P}_s \triangleq \sum_{h \geq 0, k \geq 1} (\widetilde{s}_i \setminus \square_{s_o} \setminus \square)^h \underline{s}_i \setminus \square_{s_o} \setminus \square (\widetilde{s}_i \setminus \square_{s_o} \setminus \square)^k . P_s$, if $s \notin C_{i+1}$
 (h') $\overline{P}_s \triangleq \sum_{h \geq 0} (\widetilde{s}_i \setminus \square_{s_o} \setminus \square)^h \underline{s}_i \setminus \square_{s_o} . \overline{P}_s$, if $s \notin C_{i+1}$

where, by Lemma 12.2, h is the number of reactions requiring the absence of s plus possibly some reactions requiring s ; and k is the number of reactions producing s .

It is worth nothing that, depending on the presence (P_s) or the absence (\overline{P}_s) of each entity s , for each process P_a (encoding a reaction a) the choice between the execution of the reaction code (points (a), (b), (c)) or the code expressing that reaction a is not applicable (point (d)) is deterministic. Also, the building of the code of process Cxt (points (e), (f)), is univocally determined by the evolution of γ . It derives that the trend followed by the processes P_s (or \overline{P}_s) is also deterministic (points (f), (g), (h) or (f'), (g'), (h')), leading to $P' = \llbracket \mathcal{A}, \gamma^1 \rrbracket$.

Corollary 17 (Correctness 2). *Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ and $j \geq 1$. If there exists P'' such that $P \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau}^j P''$, then letting $\pi^j = (\gamma^j, \delta^j)$ we have $P'' = \llbracket \mathcal{A}, \gamma^j \rrbracket$.*

PROOF. We proceed by induction on the transition number $j \geq 0$.

base case $j = 1$: This case falls into the case of Proposition 16.

inductive case: We assume, by inductive hypothesis, that $\exists P'$ such that

$$P \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau}^{j-1} P'$$

and $P' = \llbracket \mathcal{A}, \gamma^{j-1} \rrbracket$. As P' is the encoding of an extended interactive process, by Proposition 16, it exists P'' such that $P' \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau} P''$, and $P'' = \llbracket \mathcal{A}, \gamma^j \rrbracket$.

Proposition 18 (Completeness 1). *Let $P = \llbracket \mathcal{A}, \gamma \rrbracket$ and $\pi^1 = (\gamma^1, \delta^1)$. Then, $P \xrightarrow{\tau \setminus \tau \dots \tau \setminus \tau} P' = \llbracket \mathcal{A}, \gamma^1 \rrbracket$.*

PROOF. By Proposition 16, if there exists P' such that $P \xrightarrow{\tau \setminus \tau \cdots \tau \setminus \tau} P'$, then the structure of P' is deterministically computed.

Now, to prove that always exists P' , we observe that even in the case no reaction a is applicable in the interactive process π in A , then process P can always execute a step transition, as its subprocesses P_a can always execute one of the *alternative code for when reaction a is not applicable* (see Definition 11, code for P_a processes).

Corollary 19 (Completeness 2). *Let $P = \llbracket A, \gamma \rrbracket$ and $\pi^j = (\gamma^j, \delta^j)$. Then,*

$$P \xrightarrow{\tau \setminus \tau \cdots \tau \setminus \tau} P'' = \llbracket A, \gamma^j \rrbracket.$$

PROOF. The proof proceeds by induction on the number j , and it is similar to the one of Corollary 17.

Theorem 33 (Correspondence). $\sim_F = \equiv_{\mathcal{L}_F}$

PROOF. The proof is just an adaptation of the classical result. The two implications are proved separately.

$\sim_F \subseteq \equiv_{\mathcal{L}_F}$: Given any two processes $P \sim_F Q$ we need to prove that for any bioHML formula $G \in \mathcal{L}_F$ we have $P \models G$ iff $Q \models G$. Without loss of generality, we prove that $P \models G$ implies $Q \models G$. The proof is by structural induction on G .

- if $G = \mathbf{t}$, then $Q \models G$.
- if $G = \mathbf{f}$, then the assumption $P \models G$ is false and the implication holds.
- if $G = G_1 \wedge G_2$ we take as inductive hypotheses that

$$\begin{aligned} \forall R, S. R \sim_F S \wedge R \models G_1 &\Rightarrow S \models G_1 \\ \forall R, S. R \sim_F S \wedge R \models G_2 &\Rightarrow S \models G_2 \end{aligned}$$

We need to prove that $Q \models G$. Since $P \models G = G_1 \wedge G_2$ we have $P \models G_1$ and $P \models G_2$. Since $P \sim_F Q$, by inductive hypotheses we get $Q \models G_1$ and $Q \models G_2$. Hence $Q \models G_1 \wedge G_2 = G$.

- if $G = G_1 \vee G_2$ we take as inductive hypotheses that

$$\begin{aligned} \forall R, S. R \sim_F S \wedge R \models G_1 &\Rightarrow S \models G_1 \\ \forall R, S. R \sim_F S \wedge R \models G_2 &\Rightarrow S \models G_2 \end{aligned}$$

We need to prove that $Q \models G$. Since $P \models G = G_1 \vee G_2$ we have $P \models G_1$ or $P \models G_2$. If $P \models G_1$, since $P \sim_F Q$, by inductive hypotheses we get $Q \models G_1$ and thus $Q \models G_1 \vee G_2 = G$. If $P \models G_2$, since $P \sim_F Q$, by inductive hypotheses we get $Q \models G_2$ and thus $Q \models G_1 \vee G_2 = G$.

- if $G = \langle \chi \rangle H$ we take as inductive hypothesis that

$$\forall R, S. R \sim_F S \wedge R \models H \Rightarrow S \models H$$

860 We need to prove that $Q \models G$. Since $P \models \langle \chi \rangle H$ it means that there exists v, P' such that $P \xRightarrow{v} P'$ with $v \models \chi$ and $P' \models H$. Since $P \sim_F Q$, there exists w, Q' such that $Q \xRightarrow{w} Q'$ with $w \models \chi$ and $P' \sim_F Q'$. Then, by inductive hypothesis, $Q' \models H$ and thus $Q \models \langle \chi \rangle H = G$.

- if $G = [\chi]H$ we take as inductive hypothesis that

$$\forall R, S. R \sim_F S \wedge R \models H \Rightarrow S \models H$$

865 We need to prove that $Q \models G$. If there is no $v \models \chi$ such that $Q \xRightarrow{v} Q'$ for some Q' , then $Q \models [\chi]H = G$ trivially. For any v, Q' such that $Q \xRightarrow{v} Q'$ with $v \models \chi$, then as $P \sim_F Q$ there must exist w, P' such that $P \xRightarrow{w} P'$ with $w \models \chi$ and $P' \sim_F Q'$. Since $P \models G = [\chi]H$ then it must be $P' \models H$. Since $P' \sim_F Q'$, by inductive hypothesis $Q' \models H$. Hence $Q \models [\chi]H = G$.

870 $\equiv_{\mathcal{L}_F} \subseteq \sim_F$: We prove that $\equiv_{\mathcal{L}_F}$ is a bio-simulation and thus included in \sim_F . Take two generic processes $P \equiv_{\mathcal{L}_F} Q$ and suppose $P \xRightarrow{v} P'$ for some v, P' .

- If $v \models F$ we want to prove that there exists some w, Q' such that $Q \xRightarrow{w} Q'$, with $w \models F$ and $P' \equiv_{\mathcal{L}_F} Q'$.

875 Towards a contradiction, assume that we cannot find such w, Q' . If there is no transition $Q \xRightarrow{w} Q'$ such that $w \models F$, then the bioHML formula $G \triangleq \langle F \rangle \mathbf{t}$ is such that $P \models G$ and $Q \not\models G$, contradicting the assumption $P \equiv_{\mathcal{L}_F} Q$.

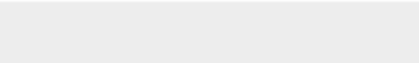

880 Otherwise, let $\mathcal{Q} \triangleq \{Q' \mid \exists w. Q \xRightarrow{w} Q' \wedge w \models F\}$ be the (non-empty) set of processes reachable from Q via a transition with a (complete) label that satisfies F . Since our processes are with guarded recursion, the set \mathcal{Q} is finite. Let $\mathcal{Q} = \{Q'_1, \dots, Q'_n\}$. By hypothesis all processes in \mathcal{Q} must not be bio-logically equivalent to P' , hence for any $i \in [1, n]$ there exists a bioHML formula $G_i \in \mathcal{L}_F$ such that $P' \models G_i$ and $Q'_i \not\models G_i$ (if it was the opposite, $P' \not\models H_i$ and $Q'_i \models H_i$ for some H_i , we can use the converse formula $G_i \triangleq \overline{H_i}$). But then the formula $G \triangleq \langle F \rangle (G_1 \wedge \dots \wedge G_n)$ is such that $P \models G$ and $Q \not\models G$, contradicting the assumption $P \equiv_{\mathcal{L}_F} Q$.


- If $v \not\models F$ then the proof is analogous to the previous case (by exploiting $\neg F$) and thus omitted.



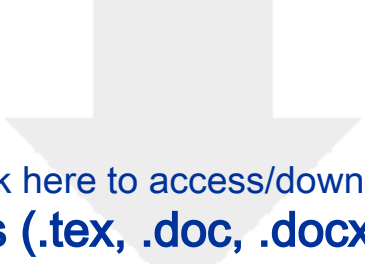
[Click here to access/download](#)

Source files (.tex, .doc, .docx, .eps, etc.)
main_TCS.tex






Click here to access/download
Source files (.tex, .doc, .docx, .eps, etc.)
09-appendix.tex





Click here to access/download
Source files (.tex, .doc, .docx, .eps, etc.)
08-conclusion.tex



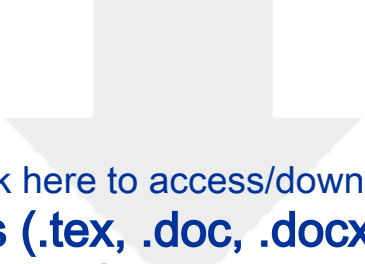


Click here to access/download
Source files (.tex, .doc, .docx, .eps, etc.)
07-discussion.tex

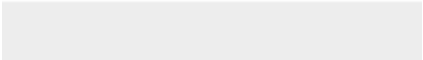





Click here to access/download
Source files (.tex, .doc, .docx, .eps, etc.)
06t-LTSforRS.tex



Click here to access/download
Source files (.tex, .doc, .docx, .eps, etc.)
06c-bioSimExample.tex





[Click here to access/download](#)

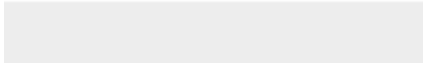
Source files (.tex, .doc, .docx, .eps, etc.)
06b-bisimulation.tex






Click here to access/download

Source files (.tex, .doc, .docx, .eps, etc.)
06-examples.tex





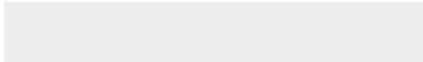
Click here to access/download
Source files (.tex, .doc, .docx, .eps, etc.)
05-translation.tex

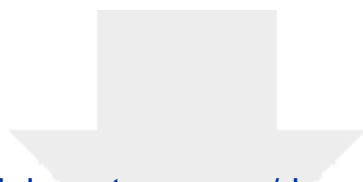




[Click here to access/download](#)

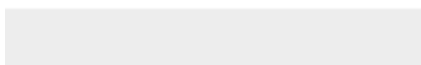
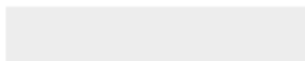
Source files (.tex, .doc, .docx, .eps, etc.)
04-CCNA.tex

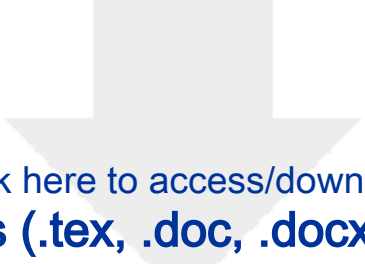





[Click here to access/download](#)

Source files (.tex, .doc, .docx, .eps, etc.)
02-ReactionSystem.tex





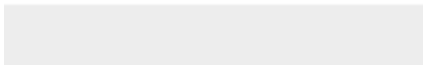
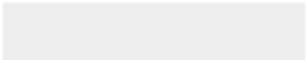
Click here to access/download
Source files (.tex, .doc, .docx, .eps, etc.)
01-introduction.tex





[Click here to access/download](#)


Source files (.tex, .doc, .docx, .eps, etc.)
00-abstract.tex





[Click here to access/download](#)


Source files (.tex, .doc, .docx, .eps, etc.)
biblio_CNA.bib



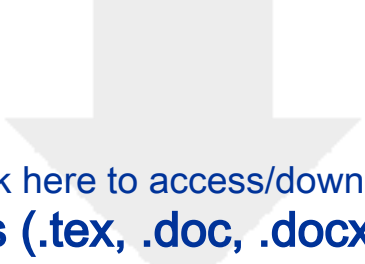
Click here to access/download
Source files (.tex, .doc, .docx, .eps, etc.)
macros.tex



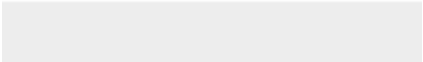

Click here to access/download
Source files (.tex, .doc, .docx, .eps, etc.)
elsarticle.cls




Click here to access/download
Source files (.tex, .doc, .docx, .eps, etc.)
elsarticle-num.bst



Click here to access/download
Source files (.tex, .doc, .docx, .eps, etc.)
elsarticle-num-names.bst





Click here to access/download
Source files (.tex, .doc, .docx, .eps, etc.)
elsaarticle-num.bst



Click here to access/download
Source files (.tex, .doc, .docx, .eps, etc.)
natbib.sty

Declaration of interests

X The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: