Fuzzy Matching in Symbolic Systems Biology

Adrián Riesco[1] Beatriz Santos-Buitrago[2] Merrill Knapp[3]
Gustavo Santos-García[4] Carolyn Talcott[3]

1. Universidad Complutense de Madrid, Madrid, Spain
2. Bio and Health Informatics Lab, Seoul National University, Seoul, South Korea
3. Computer Science Laboratory, SRI International, Menlo Park, California, USA
4. University of Salamanca, Salamanca, Spain

HSB 2019, Prague
April 7th, 2019

Motivation

- Symbolic systems biology aims to explore biological processes as whole systems instead of small and independent elements.
- The objective is to define formal models closer to the biologists mindsets.
- It is equally important to be able to compute with, analyze, and reason about these networks of biomolecular interactions at multiple levels of detail.
- Such models may suggest new insights and understanding of complex biological mechanisms.

Motivation

- Biological interactions can be handled with rule-based modeling in a natural way.
- We will focus on **Pathway Logic**, based on **Maude** formal language.
- It defines system states and rules specifying the manners in which the state changes along time.
- It supports many kinds of analysis: simulation, search, model checking, and meta-analysis.

Motivation: Terms

- The **basic data structure** in Pathway Logic is a set of *locations*.

  ```
  {L | S} {L' | S'} ...
  ```

- Each location is a pair with the location identifier and a *soup* of elements in the location.

  ```
  [E - M] E' ...
  ```

- For example,

  ```
  {CLc | [Mek1 - act] [Erk1 - act phos(Y 204)] }
  {NUc | Maz [Tp53-gene - on] Rb1 Chek1 Chek2 Myc Tp53 NProteasome}
  ```

Motivation: Rules

- Rules are usually unconditional.
- They are composed of reactants, products, and controls:
  - *Controls* are the occurrences that appear in the rule's premise and conclusion.
  - *Reactants* are the occurrences that appear in the premiss but not the conclusion.
  - *Products* are the occurrences that appear in the conclusion but not the premise.

Motivation: Rules

- Rules have the following form:

```
rl [632c.Akts.by.Ilk]:
   {CLc | clc [Ilk - act] Akts}
=> {CLc | clc [Ilk - act] [Akts - phos(FSY)]} .
```

- With the following meaning:
  - `CLc` corresponds to the cytoplasm.
  - The protein `Ilk` is activated.
  - A protein in the `Akts` family is found.
  - The variable `clc` matches the rest of the elements in the set.
  - After the reaction `Akts` will be phosphorylated on `FSY`.

Motivation: Rules

- The rule

```
rl [632c.Akts.by.Ilk]:
   {CLc | clc [Ilk - act] Akts}
=> {CLc | clc [Ilk - act] [Akts - phos(FSY)]} .
```

can be understood as:

```
crl [632c.Akts.by.Ilk]:
    {CLc | clc Akts}
 => {CLc | clc [Akts - phos(FSY)]}
 if clc' [Ilk - act] := clc .
```

Motivation: Modifications

- Some aspects might be more or less general depending on the rule.
- For example, we can use the `Akts` family or a specific member, like `Akt1`.
- We can indicate the reaction is phosphorylated (`phos(FSY)`) or phosphorylated in a particular location (`phos(Y 123)`).

Motivation

- The point now is: **how to build a model from some data?**
- Starting from an initial state, the Pathway Logic model is obtained by a symbolic reasoning process to derive instances of rules.
- These instances are derived from a rule knowledge base relevant to the given initial state.
- **Why instances?**
- Because the reactions are, in some cases, slightly different from the particular rules.

Motivation

- Some examples of what can go wrong are:
  - The state contains `[Akts - act phos]` and the rule premise requires `[Akts - phos]`.
  - A rule requires `Akts` (the family) to be present and the state contains `Akt1` (a member of the family).
  - The state contains `[Akts - Yphos]` and the rule premise requires `[Akt - phos]`.
- In the last two cases the rule is "**more general**" than the state, so it should match somehow.
- However, the notion of generality is not the one used in Maude language, involving variables of a given sort and ground terms.

Motivation: Families and Products

- Note that we have information about **families**:

```
op Akts : -> AktS [ctor metadata "(\
  (type Family)\
  (members Akt1 Akt2 Akt3))"] .
```

- As well as for the corresponding **products**:

```
op FSY :  -> Site [ctor metadata "(\
  (FamilyName Akts)\
  (epitopes \"Akt1 phos(S473)\"\
            \"Akt2 phos(S474)\"\
            \"Akt3 phos(S472)\"))"] .
```

Motivation

- Maude standard matching mechanism cannot deal with these problems.
- Thus far this was solved by hand.
- So we propose a notion of *fuzzy matching* to deal with them.
- This matching has been implemented at the metalevel and integrated with Pathway Logic via Full Maude.
- The current version does not really implement matching; it adds generic rules that will work with standard matching.

Fuzzy matching: Forwards and backwards analysis

- We perform *forwards* and *backwards* analysis.
- Forwards analysis works from initial states.
- Backwards analysis works from reached states.

Fuzzy matching

- We first define a notion of `P > P'`, read "`P` is more general than `P'`."
- Generality is measured with respect to the knowledge base.
- For example `phos > Yphos > phos(Y 123)`, where
  - `phos` says phosphorylation on some site.
  - `Yphos` says phosphorylation on a tyrosine site.
  - `phos(Y 123)` says phosphorylation a tyrosine site at position 123.
- For simplicity, `P > P'` also holds if `P = P'`.

Fuzzy matching

- This notion is easily extended to locations.
- In this case we have

```
{L | [P - mods]} >> {L | [P' - mods']}
```

- It holds if `P > P'` and $\mathtt{m_i} > \mathtt{m'_i}$, with $\mathtt{m_i}$ an element of `mods` and $\mathtt{m'_i}$ an element of `mods'`.

Forward matching

- Given a rule of the form

```
rl {L | CONT [P - mods mods0]}
=> {L | CONT [P - mods mods1]}
   if CONT' [Q - qmods] := CONT .
```

  where `mods` and/or `qmods` may be empty, and `mods0` or `mods1` may be empty but not both.

- And a state

```
{L | [Q' - qmods'] [P' - mods' mods0] }
```

Forward matching

- The rule fuzzy forward matches the term if
  `{L | [P - mods]} >> {L | [P' - mods']}`
  and
  `{L | [Q - qmods]} >> {L | [Q' - qmods']}`
- In this case, the result of applying the rule will be
  `{L | [Q' - qmods'] [P' - mods' mods1]}`

Forward matching

- In our previous example, given the rule

```
crl [632c.Akts.by.Ilk]:
    {CLc | clc Akts}
 => {CLc | clc [Akts - phos(FSY)]}
 if clc' [Ilk - act] := clc .
```

- It can be fuzzily applied to the state

```
{CLc | clc Akt1 [Ilk - act]}
```

- Obtaining as result

```
{CLc | clc [Akt1 - phos(FSY)]}
```

Backwards matching

- Given a reached state, we can also use fuzzy matching to collect information about the source state.
- In this case we have the same rule schema:

```
crl {L | CONT [P - mods mods0]}
 => {L | CONT [P - mods mods1]}
    if CONT’ [Q - qmods] := CONT .
```

- And the collected state:

```
{L | [P’ - mods’ mods1] }
```

Backwards matching

- The rule fuzzy backwards matches the term if

  `{L | [P - mods]} >> {L | [P' - mods']}`

- And the source term will be

  `{L | [P' - mods' mods0] [Q - qmods]}`

Backwards matching

- Assume we want to reach `[Eif4ebp1 - phos(S 65)]` in `CLc`.
- The state contains `[Akts - act phos(FSY) phos(KTF)]` in the `CLc`.
- In this case the rule `819c` fuzzy matches by refining the modifications in `Akts` to match the available occurrence:

```
rl [819c.Eif4ebp1.by.Akts]:
   {CLc | clc [Akts - act]  Eif4ebp1}
=> {CLc | clc [Akts - act] [Eif4ebp1 - phos(S 65)] } .
```

Implementation

- We have implemented fuzzy matching as an extension of Full Maude language.
- This extension allows users to perform some analysis that the standard Pathway Logic implementation does not support.
- In particular, it supports *narrowing*.
- It also allows us to manipulate the modules in the database.
- However, these changes are not integrated with the Pathway Logic Assistant.

Implementation

- The current implementation does not really implement a matching function.
- It instantiates the rules so the Maude standard matching algorithm.
- This simple approach allows us to evaluate the usefulness of fuzzy matching.
- However, it presents a bad performance:
  - The transformation itself takes time.
  - The obtained module is huge in general, so the execution is also slow.

Conclusions

- We proposed a notion of fuzzy matching that captures the fuzziness of experimental results.
- It allows rules to be matched by specializing or generalizing occurrences to match.
- We presented a prototype that implements this notion of matching.

Ongoing work

- The current implementation of the system modifies the rules in the module so Maude standard matching works as fuzzy matching.
- This has the advantage of producing all possible concrete rules.
- However, it also generates very large models and many irrelevant rules.
- Our aim is to instantiate by need, starting with an initial state of interest.
- Once implemented, we want to integrate it with the Pathway Logic Assistant.
- Finally, we want to apply the algorithm to relevant examples and analyze how it behaves.

Fuzzy Matching in Symbolic Systems Biology

Adrián Riesco[1] Beatriz Santos-Buitrago[2] Merrill Knapp[3]
Gustavo Santos-García[4] Carolyn Talcott[3]

1. Universidad Complutense de Madrid, Madrid, Spain

2. Bio and Health Informatics Lab, Seoul National University, Seoul, South Korea

3. Computer Science Laboratory, SRI International, Menlo Park, California, USA

4. University of Salamanca, Salamanca, Spain

HSB 2019, Prague
April 7th, 2019