# Responsive Choice in Mobile Processes

Maxime Gamboni and António Ravara

May 25, 2009

## Responsiveness

In the $\pi$-calculus [MPW92], *activeness* is the ability to get in contact with a process through a particular channel, and *responsiveness* is the ability to conduct a conversation, or session, through that channel. The latter is of course strongly dependent on the *type* of the channel. These have been widely studied concepts, dating back from Sangiorgi's *receptiveness* [San99], and closely related to Acciai and Boreale's *responsiveness* [AB08] or Kobayashi's *livelock freedom* [Kob02]. A common approach is to ensure there are no *livelocks* by assigning *levels* to name occurrences, effectively calculating in which order they become active.

We are particularly interested in *conditional* activeness, specifically in statements such as $\gamma \triangleleft \varepsilon$, where $\gamma$ and $\varepsilon$ are logical statements on channel activeness, and the whole meaning "$\gamma$ holds *provided* $\varepsilon$ is made available (e.g. through parallel composition)". In terms of levels this could be thought as an inequality whose unknowns are the channel levels. This permits analysing several components of a system individually, and then predict the behaviour of the whole by *composing* the components' types.

## Choices

In this work we integrate the concept of *choice* to activeness and responsiveness.

Choices are a fundamental way of representing data in the basic $\pi$-calculus (although there are extensions that include data such as numbers as primitives). For example, Milner's encoding of Boolean values represents Boolean values as receivers on two parameter channels: $\mathsf{True}$ replies to queries with a signal on the first parameter ($!\, b(tf).\bar{t}$) and $\mathsf{False}$ with a signal on the second parameter ($!\, b(tf).\bar{f}$). We say those two processes are instances of (internal) *choice* because they "decide" to send a signal to one of the parameters rather than to the other. A $\mathsf{Random}$ Boolean could be implemented as $!\, b(tf).(\boldsymbol{\nu} x)\,(\bar{x} \mid (x.\bar{t}{+}x.\bar{f}))$, in which the choice is performed "at run-time" by the sum ("+").

A choice made by one process may cause *branching* (also called external choice) in another process.

Branching is typically implemented with the $\pi$-calculus sum operator, as in $\bar{b}(\boldsymbol{\nu} tf).(t.P{+}f.Q)$, which executes $P$ if $b$ is $\mathsf{True}$, and $Q$ if $b$ is $\mathsf{False}$. For example, the "$r = a$ $\mathsf{and}$ $b$" logical circuit can be implemented as follows:

$$!\, r(tf).\bar{a}(\boldsymbol{\nu} t'f').(t'.\bar{b}\langle tf\rangle{+}f'.\bar{f}), \tag{1}$$

We first query $a$. If it returns true ($t'$) then $r$ returns the same as $b$. If it returns false instead, $r$ itself returns false ($f$). So, depending on $a$ and $b$'s behaviour, either a signal will either be sent on $t$, or one will be sent on $f$ (but never both).

To the best of our knowledge, no existing work deals with this kind of processes. The usual approach of assigning a single numerical level to name occurrences would not work well here because, as $t$ may never get triggered (in case $r$ returns false), it would require an infinite level, and similarly $f$ would require an infinite level as well. We need a typing system able to capture the fact that *exactly one* of $\bar{t}$ and $\bar{f}$ will eventually get triggered when $r$ is queried. Kobayashi's Generic Type System [IK01] does include choice into its types but is targeted on *safety* properties such as deadlock-freedom or race-freedom.

## Our Approach

In contrast to level-based analysis, dependency-based systems as we have explored in the past

[GR09] are naturally expanded with choice and branching operators.

We chose to focus on choice itself, leaving out features like recursivity [AB08] and complex channel usages such as locks [Kob02], which have already been well studied.

*Channel Types* describe, both for the input and output halves, which capabilities (input or output) of the parameters may be used, and what are the activeness requirements. In addition they specify what choices may be done using a logical disjunction operator "$\vee$", and what branching they must offer, using the branching operator "$+$".

The input side of a Boolean channel has type

$$(\bar{1}_{\mathbf{A}}, \bar{2}^0) \ \vee \ (\bar{1}^0, \bar{2}_{\mathbf{A}}), \tag{2}$$

that says that either the first parameter ("1") must be output ("$\bar{1}$") active ("$_{\mathbf{A}}$"), and the second parameter unused ("$\bar{2}^0$"), or ("$\vee$") the opposite (note that the "$\vee$" operator is exclusive).

The output side has type

$$1_{\mathbf{A}} + 2_{\mathbf{A}}, \tag{3}$$

which has a similar meaning, but on the input side of the parameters, and with the additional constraint that inputs at the parameters ("1" and "2") must be the guards of a sum ("$+$").

A Boolean channel is now said input (resp., output) responsive if its input (resp., output) halves respect the protocol outlined above.

*Process Types* are similar to channel types, but refer to actual channel names rather than parameter numbers. They also specify free names' channel types but we omit them here for space reasons. It is now easy to see that (1) can be described with the statement $r_{\mathbf{R}} \triangleleft (a_{\mathbf{AR}} \wedge b_{\mathbf{AR}})$ ($r$ is responsive if both $a$ and $b$ are active and responsive). If the input on a channel is active and responsive then it will behave according to the protocol specified in the channel type whenever queries are sent to it. Formally, for the $\bar{b}\langle tf \rangle$ process, this is written $(\bar{t}_{\mathbf{A}} \vee \bar{f}_{\mathbf{A}}) \triangleleft b_{\mathbf{AR}}$, where the left side is just (3) with $t$ and $f$ replacing 1 and 2 (omitting terms with a zero exponent).

A sum $t'.T + f'.F$ is considered to have the type $(t'_{\mathbf{A}} + f'_{\mathbf{A}}) \wedge (\Gamma_T \triangleleft \bar{t'}_{\mathbf{A}} \vee \Gamma_F \triangleleft \bar{f'}_{\mathbf{A}})$ (where $\Gamma_T$ and $\Gamma_F$ are respectively the types of $T$ and $F$): The process offers a branching $t' + f'$ and, in addition ("$\wedge$") chooses ("$\vee$") one of $\Gamma_T$ and $\Gamma_F$ depending respectively on a message on $t'$ and on one sent on $f'$. The decoupling between the guards and the continuations is done to make explicit which channels must be used to make the process branch. In the example (1) the bracketed portion $(t'.\bar{b}\langle tf \rangle + f'.\bar{f})$ has type $(t'_{\mathbf{A}} + f'_{\mathbf{A}}) \wedge ((\bar{t}_{\mathbf{A}} \vee \bar{f}_{\mathbf{A}}) \triangleleft b_{\mathbf{AR}} \vee \bar{f}_{\mathbf{A}})$.

Adding the request to $a$ breaks the branching (and subsequently hides $t'$ and $f'$ as they are bound), and this becomes

$$(\bar{t}_{\mathbf{A}} \vee \bar{f}_{\mathbf{A}}) \triangleleft (b_{\mathbf{AR}} \wedge a_{\mathbf{AR}}). \tag{4}$$

Now $r$ is responsive if it behaves according to the protocol given in (2), which is written $r_{\mathbf{R}} \triangleleft (\bar{t}_{\mathbf{A}} \vee \bar{f}_{\mathbf{A}})$. Composing with (4) *collapses* the dependency chain and we obtain $r_{\mathbf{R}} \triangleleft (b_{\mathbf{AR}} \wedge a_{\mathbf{AR}})$, as required.

# Conclusion

In this short presentation we outlined our current research on integrating *choice* and *responsiveness*, which permits statically guaranteeing responsiveness on process constructs that are necessary for encoding data, but that could not be analysed by existing works. Moreover, *conditional properties* permit analysing components of a system individually. We are developing a general type notation, formal semantics and a sound type system which are able to recognise most common kinds of data encodings such as Booleans, numbers, lists, and so on, as well as any non-recursive function on them.

# References


[AB08] L. Acciai and M. Boreale. Responsiveness in process calculi. *Theoretical Computer Science*, 409(1):59–93, 2008.

[GR09] M. Gamboni and A. Ravara. Activeness and Responsiveness in Mobile Processes. In *7th Conference on Telecommunications*, pages 429–432. Instituto de Telecomunicações, 2009.

[IK01] A. Igarashi and N. Kobayashi. A generic type system for the Pi-calculus. *ACM SIGPLAN Notices*, 36(3):128–141, 2001.

[Kob02] N. Kobayashi. A type system for lock-free processes. *Information and Computation*, 177(2):122–159, 2002.

[MPW92] R. Milner, J. Parrow and D. Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100(1):1–77, 1992.

[San99] D. Sangiorgi. The Name Discipline of Uniform Receptiveness. *Theoretical Computer Science*, 221(1–2):457–493, 1999.