Foreword

With warm-hearted and friendly promotion by our Japanese friends Prof. Atsushi Ohori, Prof. Tetsuo Ida, and Prof. Zhenjiang Hu, and other distinguished professors and scholars from countries and regions such as Japan, South Korea, Singapore, and Taiwan, the 1st Asian Symposium on Programming Languages and Systems (APLAS 2003) took place in Beijing. We received 76 papers, among which 24 were selected for the proceedings after serious evaluation, which fully demonstrates the high quality of the collected papers. I hereby, on behalf of the Program Committee and the Organization Committee of the symposium, would like to extend the warmest welcome and hearty thanks to all colleagues who attended the symposium, all scholars who generously contributed their papers, and all those who were actively dedicated to the organization of this symposium.

Over the past decade, the Asian economy has undergone rapid development. Keeping pace with this accelerated economic growth, Asia has made great headway in software, integrated circuits, mobile communication and the Internet. All this has laid a firm material foundation for undertaking theoretical research on computer science and programming languages. Therefore, to meet the increasing demands of the IT market, great opportunities and challenges in advanced research in these fields. I strongly believe that in the coming future, with the persistent efforts of our colleagues, the Asian software industry and research on computer science will be important players in the world economy, on an equal footing with their counterparts in the United States and Europe.

I am of the opinion that, to enhance Asian computer science research, much more attention should be paid to the new issues and technical problems brought with the development of the software industry in the world, particularly in Asia, and accordingly, we should advance computer science research in Asia to a position with distinctive features and great prospects. In the course of the Asian economic development over the past 10 years, the pursuit of highly reliable software and efficient software development processes has created urgent demands for basic research on computer science. In addition, theoretical guidelines are required to solve the problems of congestion, waste and security generated in the storage, transmission and processing of massive information on the Internet. Under such circumstances, it is expected that a new discipline, namely Information Physics, will be born in the near future.

Dear colleagues, as leading theorists of computer science in Asia, we should take up this task and put all our efforts into achieving creative breakthroughs in the fields mentioned above, and promoting our ongoing contacts and cooperation with our European and American colleagues, and thus turn Asia into a promising land of research on computer science and technology, and make a historical contribution to the progress of mankind as a whole.

September 2003

Wei Li
Preface

This volume contains the proceedings of APLAS 2003, the 1st Asian Symposium on Programming Languages and Systems, held in Beijing, China, November 27–29, 2003, sponsored by the Asian Association for Foundation of Software (AAFS) and Beihang University.

The symposium was devoted to foundational issues in programming languages and systems, covering (but not limited to) the following areas:

– concurrency and parallelism,
– language implementation and optimization,
– mobile computation and security,
– program analysis and verification,
– program transformation and calculation,
– programming paradigms and language design,
– programming techniques and applications,
– semantics, categorical and logical foundations,
– tools and environments, and
– type theory and type systems.

In response to the call for papers, 76 papers were submitted by authors from Australia, Austria, China, Denmark, France, India, Italy, Japan, Korea, Portugal, Singapore, Spain, Taiwan, the United Arab Emirates, the UK, and the USA. Each paper was reviewed by at least three program committee members with the help of external expert reviewers. The program committee meeting was conducted electronically from August 5 through August 15th. The competition was very tough and the deliberation was a difficult process. After careful and thorough discussion, the program committee selected 24 papers. I would like to sincerely thank all the members of the APLAS 2003 Program Committee for their excellent job, and all the external reviewers for their invaluable contribution. The submission and review process was managed using the CyberChair system.

In addition to the 24 contributed papers, the symposium also included talks by three invited speakers: Thomas A. Henzinger (University of California at Berkeley, USA), Simon Peyton Jones (Microsoft Research, UK), and Wen-tsun Wu (Academia Sinica, China). I am grateful to the three invited speakers.

Many people helped in the effort to establish this new Asian-based conference series, APLAS, as a high-quality forum to serve the worldwide programming languages community. Without their help and enthusiastic support, APLAS 2003 would not have happened. My special thanks to our general chair, Wei Li, whose initiative and efforts made the first APLAS in Beijing possible. I would like to thank Shilong Ma. In addition to his hard work as a program committee member, he helped us to solve numerous problems we encountered during our planning, and during APLAS 2003, itself. The AAFS steering committee provided advice and suggestions. I would particularly like to thank Tetsuo Ida, who provided advice and suggestions at several critical moments.

September 2003

Atsushi Ohori
Organization

General Chair
Wei Li
Beihang University, China

Program Chair
Atsushi Ohori
Japan Advanced Institute of Science and Technology, Japan

Program Committee
Manuel Chakravarty
University of New South Wales, Australia
Wei Ngan Chin
National University of Singapore, Singapore
Tyng-Ruey Chuang
Academia Sinica, Taiwan
Yuxi Fu
Shanghai Jiaotong University, China
Masahito Hasegawa
Kyoto University, Japan
Kohei Honda
Queen Mary and Westfield College, UK
Zhenjiang Hu
University of Tokyo, Japan
Peter Lee
Carnegie Mellon University, USA
Shilong Ma
Beihang University, China
Martin Odersky
École Polytechnique de Lausanne, Switzerland
Don Sannella
University of Edinburgh, UK
Zhong Shao
Yale University, USA
Kwangkeun Yi
Seoul National University, Korea
Taiichi Yuasa
Kyoto University, Japan

Referees
Philippe Altherr
Florin Craciun
Hugh Anderson
Vincent Cremet
Stefan Andrei
Pascal Cuoq
David Aspinall
Neil D. Jones
Martin Berger
Kyung-goo Doh
Lennart Beringer
Burak Emir
Andrew Bernard
Kai Engelhardt
Johannes Borgstroem
Hyunjoon Eo
Patricia Bouyer
Lin Gao
Chiyun Chen
Maria Garcia de la Banda
Kwanghoon Choi
Samir Genaim
William D Clinger
Stephen Gilmore
Table of Contents

Invited Talk 1
On a Method of Global Optimization .............................................. 1
   Wen-tsun Wu

Session 1
Observing Asymmetry and Mismatch .............................................. 2
   Xiaoju Dong, Yuxi Fu
Expressive Synchronization Types for Inheritance
in the Join Calculus .......................................................... 20
   Qin Ma, Luc Maranget
Term Graph Rewriting for the $\pi$-Calculus .................................. 37
   Fabio Gadducci
Useless-Code Elimination and Program Slicing for the Pi-Calculus ...... 55
   Naoki Kobayashi

Session 2
Constraint Functional Logic Programming for Origami Construction ....... 73
   Tetsuo Ida, Mircea Marin, Hidekazu Takahashi
A Typeful and Tagless Representation for XML Documents ................. 89
   Dengping Zhu, Hongwei Xi
Dataflow Pointcut in Aspect-Oriented Programming .......................... 105
   Hidehiko Masuhara, Kazunori Kawauchi

Session 3
Affine-Based Size-Change Termination ......................................... 122
   Hugh Anderson, Siau-Cheng Khoo
Using Locality of Flow in Dataflow Analyses .................................. 141
   Ravindra D V
A Differential Fixpoint Evaluation Framework
for Non-distributive Systems ................................................... 159
   Joonseon Ahn, Yong Jin Kwon
Invited Talk 2
Model Checking: From Hardware to Software .......................... 176
Thomas A. Henzinger

Session 4
Executing Verified Compiler Specification ............................. 178
Koji Okuma, Yasuhiko Minamide

Controlling and Optimizing the Usage of One Resource .............. 195
Antoine Galland, Mathieu Baudet

Resource Usage Verification ............................................. 212
Kim Marriott, Peter J. Stuckey, Martin Sulzmann

Automatic Construction of Hoare Proofs from Abstract Interpretation Results .................................................. 230
Sunae Seo, Hongseok Yang, Kwangkeun Yi

Session 5
A Program Inverter for a Functional Language with Equality and Constructors ....................................................... 246
Robert Glück, Masahiko Kawabe

Rebuilding a Tree from Its Traversals: A Case Study of Program Inversion 265
Shin-Cheng Mu, Richard Bird

Type Inference with Structural Subtyping: A Faithful Formalization of an Efficient Constraint Solver ................. 283
Vincent Simonet

Session 6
Continuation Semantics for Parallel Haskell Dialects. ................. 303
Mercedes Hidalgo-Herrero, Yolanda Ortega-Mallén

Translating a Continuous-Time Temporal Logic into Timed Automata ... 322
Guangyuan Li, Zhisong Tang

The Semantic Layer of Timber .......................................... 339
Magnus Carlsson, Johan Nordlander, Dick Kieburtz

Invited Talk 3
Scrap Your Boilerplate ................................................... 357
Simon Peyton Jones, Ralf Lämmel
Session 7

Correctness of a Higher-Order Removal Transformation through a Relational Reasoning ............................................ 358
  Susumu Nishimura

Extensional Polymorphism by Flow Graph Dispatching ................. 376
  Jun Furuse

Register Allocation Based on a Reference Flow Analysis ............... 394
  Dae-Hwan Kim, Hyuk-Jae Lee

Lazy Stack Copying and Stack Copy Sharing for the Efficient Implementation of Continuations ........................................ 410
  Tomoharu Ugawa, Nobuhisa Minagawa, Tsuneyasu Komiya,
  Masahiro Yasugi, Taiichi Yuasa

Author Index .......................................................... 421
On a Method of Global Optimization
Invited Talk

Wen-tsun Wu

Institute of Systems Science, Academia Sinica, Beijing 100080, China

The usual optimization problem, in eventually introducing some further new variables, may be put in the following form: **Optimization Problem O.** Let \( f, g_j, (j = 1, \ldots, m) \) be infinitely differentiable real functions in variables \( X = \{x_1, x_2, \ldots, x_n\} \) in a domain \( D \) of \( \mathbb{R}^n(X), \mathbb{R} \) being the real field. To find **optimal** (maximum or minimum) value(s) of \( f \) for \( X \in D \) under the constraint conditions

\[
g_j = 0, j = 1, 2, \ldots, m.
\]

Various known methods for the solving of Problem O give usually only optimal values of **local** character which are in general not unique. On the other hand if the functions \( f, g_j \) are all **polynomial** ones, then we have discovered some method which will give the **unique global** optimal (greatest or least) value of the Problem, so far it is known to exist. In fact, we have the following

**Theorem.** For the Optimization Problem O with \( f, g_j \) all **polynomials** there is a **finite** set of real values \( K \) such that: If the **global** optimal (greatest or least) value of \( f \) in \( D \) does exist, then this optimal value is equal to the optimal (greatest or least) value of the **finite** set \( K \).

We call the finite set \( K \) in the theorem the **Finite Kernel Set** of Problem O. It can be algorithmically determined by means of some package due to D.K.Wang of our Institute. Various kinds of problems like non-linear programming, geometric and trigonometric inequalities-proving, definiteness of polynomials, traffic control, etc. have been solved by means of the above method.
Observing Asymmetry and Mismatch

Xiaoju Dong and Yuxi Fu*

Department of Computer Science, Shanghai Jiaotong University, Shanghai 200030, China
\{dong-xj,fu-yx\}@cs.sjtu.edu.cn

Abstract. The chi calculus is studied in the framework incorporating two constructions widely useful in applications: asymmetric communication and mismatch condition. The barbed bisimilarity is used to give a general picture of how the two constructions affect the observational theory. Both the operational properties and the algebraic properties of the enriched calculus are investigated to support an improved understanding of the bisimulation behaviors of the model.

1 Introduction

We study the asymmetric $\chi$-calculus with mismatch. The background of this investigation is as follows:

− The $\chi$-calculus was independently proposed by Fu in [2,3] and by Parrow and Victor in [15]. Fu was motivated to simplify the $\pi$-calculus and to provide a proof theoretical interpretation of concurrent computation ([5]). Parrow and Victor shared the motivation on simplification. They were also trying to give a more direct interpretation of concurrent constraint programming ([17]). The model they proposed, called Fusion Calculus, is actually the polyadic $\chi$-calculus. The $\chi$-calculus is symmetric in the sense that the input and output prefixes are of the same form and that communications are symmetric. Theoretical investigations on Fusion Calculus are carried out in [16]. More fundamental studies on $\chi$-calculus are done in [4,6,8]. The important results in [4,6,8] are the introduction of $L$-bisimilarities, the classification of $L$-bisimilarities using bisimulation lattice, and the characterizations of $L$-bisimilarities in terms of open style bisimilarities.

− Symmetry of communications is a beautiful thing to have in theory. In programming practice however there is a need for asymmetric communications. The symmetry introduces extra uncertainty. This nondeterministic feature

* The author is supported by the National Distinguished Young Scientist Fund of NNSFC (60225012), the Shanghai Science and Technology Development Fund (025115032), the Young Scientist Research Fund and the University Scholar Funding Scheme. He is also supported by BASICS, Center of Basic Studies in Computing Science, sponsored by Shanghai Education Commission. BASICS is affiliated to the Department of Computer Science of Shanghai Jiaotong University.
is not always welcome by users. In reality we often come across with situations in which we favor a command that always acts as an output instruction in whatever environments it resides. In other words we really need a command set that draws a line between an input command and an output command. Therefore it is necessary to pay as much attention to the asymmetric $\chi$-calculus as to the symmetric one. Parrow and Victor have studied the asymmetric $\chi$-calculus, which they called Update Calculus, in [14]. A more systematic investigation into the asymmetry is reported in [7]. It turns out that the theory of the asymmetric $\chi$-calculus is much richer than the theory of the symmetric $\chi$-calculus. The barred bisimilarity of the asymmetric $\chi$-calculus for example reveals some aspects of process identity for the first time. Some of the equivalence properties are beyond our intuition at first sight. But they are very reasonable from an observational viewpoint.

- A familiar programming construct is the binary $if \ B \ then \ C_1 \ else \ C_2$ command. One could say that this command is behind all the ‘intelligent behaviors’ of computer programs. In calculi of mobile processes the command $if \ B \ then \ C_1 \ else \ C_2$ can be coded up by a process like $[x=y]P + [x\neq y]Q$. Here the mismatch plays a crucial role. Now the binary command $if \ B \ then \ C_1 \ else \ C_2$ could be simulated by two consecutive unary commands $if \ B \ then \ C_1; \ if \ \neg B \ then \ C_2$. The negation operator appeared here implies that if we would translate the program $if \ B \ then \ C_1; \ if \ \neg B \ then \ C_2$ into a process in some calculus of mobile processes we would have to use the mismatch. One could argue that the positive match operator $[x=y]$ suffices in a lot of cases. This is true at the expanse that programs so constructed are unnecessarily complex since a lot of coding-up is applied. It should be pointed out that these cases are restricted to the applications where computation flows depend on enumerations of finite numbers of names. Of course the calculus can be extended with other constructs like infinite sum. However that would give rise to a model more abstract and less real. Motivated by applications, the mismatch operator turns out to be very important in theory. Some of the results in process algebra are valid only in the presence of this operator. In the framework of the $\chi$-calculus, the mismatch operator has been studied in [9,10,11]. It is demonstrated in these papers that the mismatch could be extremely subtle. Even the definitions of bisimulations call for extra care! The hyperequivalence studied in [16] for example is not even observational in the presence of mismatch. See [11] for counter examples.

In this paper we study the interplay between the asymmetry and the mismatch in the framework of the $\chi$-calculus. Both features are motivated by applications. And both are significant in theory. Based upon previous work on the asymmetry and the mismatch, we will show, both operationally and algebraically, how one-step actions of the asymmetric $\chi$-calculus with mismatch are simulated by observationally equivalent sequences of actions. We will achieve this by looking at the barbed bisimilarity. We omit the proofs and some intermediate lemmas. One could find a detailed account in the full paper [1]. In the rest of the paper we will refer to ‘asymmetric $\chi$-calculus with mismatch’ as ‘$\chi$-calculus’.
2 Asymmetric Chi Calculus with Mismatch

Let $\mathcal{N}$ be the set of names ranged over by small case letters and $\bar{\mathcal{N}}$ the set \{\(\bar{x} \mid x \in \mathcal{N}\}\} of co-names. The Greek letter $\alpha$ ranges over $\mathcal{N} \cup \bar{\mathcal{N}}$, where the dot $\cdot$ denotes set union. For $\alpha \in \mathcal{N} \cup \bar{\mathcal{N}}$, let $\bar{\sigma}$ be defined as $\alpha$ if $\alpha = \bar{\sigma}$ and as $\sigma$ if $\alpha = \bar{\sigma}$. The $\chi$-processes are defined by the following abstract grammar:

\[
P := 0 | \alpha x. P | P | P | (x)P | [x=y]P | [x \neq y]P | P + P
\]

Let $\lambda$ range over the set of transition labels \{ax, $\bar{\alpha}x$, $\alpha(x)$, $\bar{\alpha}(x)$, $y/x$, $y/x$ | $a, x, y \in \mathcal{N}\} \cdot \{\tau\}$ and $\gamma$ over \{ax, $\bar{\alpha}x$, $\alpha(x)$, $\bar{\alpha}(x)$ | $a, x, y \in \mathcal{N}\} \cdot \{\tau\}$. In $y/x$ and $y/x$, $x$ and $y$ must be different. The free and bound names are defined as usual. We use the notation $fn(\_)$, $bn(\_)$ and $n(\_)$ in their standard meaning. We write $P\{y/x\}$ for the result of substituting $y$ for $x$ throughout $P$. We will assume that a bound name is distinct from any other name in all environments.

In the following labelled transition system, symmetric rules have been systematically omitted.

**Sequentialization**

\[
\frac{}{\alpha x. P \xrightarrow{\alpha x} P}
\]

**Composition**

\[
\frac{P \xrightarrow{\gamma} P'}{P | Q \xrightarrow{\gamma} P' | Q}
\]

\[
\frac{P \xrightarrow{y/x} P'}{P | Q \xrightarrow{y/x} P' | Q\{y/x\}}
\]

\[
\frac{P \xrightarrow{(y)/x} P'}{P | Q \xrightarrow{(y)/x} P' | Q\{y/x\}}
\]

**Communication**

\[
\frac{P \xrightarrow{a(x)} P' \quad Q \xrightarrow{\bar{\alpha}y} Q'}{P | Q \xrightarrow{\tau} P'\{y/x\} | Q'}
\]

\[
\frac{P \xrightarrow{a(x)} P' \quad Q \xrightarrow{\bar{\alpha}(x)} Q'}{P | Q \xrightarrow{\tau} (x)(P' | Q')}
\]

\[
\frac{P \xrightarrow{\alpha x} P' \quad Q \xrightarrow{\bar{\alpha}x} Q'}{P | Q \xrightarrow{\tau} P' | Q'}
\]

\[
\frac{P \xrightarrow{y/x} P' \quad Q \xrightarrow{\bar{\alpha}y} Q'}{P | Q \xrightarrow{\tau} P'\{y/x\} | Q'\{y/x\}}
\]

\[
\frac{P \xrightarrow{a(x)} P' \quad Q \xrightarrow{\bar{\alpha}(y)} Q'}{P | Q \xrightarrow{\tau} P'\{y/x\} | Q'\{y/x\}}
\]

**Localization**

\[
\frac{P \xrightarrow{\lambda} P' \quad x \notin n(\lambda)}{(x)P \xrightarrow{\lambda} (x)P'}
\]

\[
\frac{P \xrightarrow{\alpha x} P' \quad x \notin \{\alpha, \bar{\alpha}\}}{(x)P \xrightarrow{\alpha x} P'}
\]

\[
\frac{P \xrightarrow{y/x} P' \quad x \notin n(\lambda)}{(x)P \xrightarrow{\tau} P'}
\]

\[
\frac{P \xrightarrow{y/x} P' \quad x \notin n(\lambda)}{(y)P \xrightarrow{\tau} (y)P'}
\]

**Condition and Selection**

\[
\frac{P \xrightarrow{\lambda} P' \quad x = x}{[x=x]P \xrightarrow{\lambda} P'}
\]

\[
\frac{P \xrightarrow{\lambda} P' \quad x \neq y}{[x \neq y]P \xrightarrow{\lambda} P'}
\]

\[
\frac{P \xrightarrow{\lambda} P' \quad P + Q \xrightarrow{\lambda} P'}{P}
Here are some examples of communication admissible by the operational semantics:

\[(x)(R \mid (\overline{a}y.P \mid ax.Q)) \overset{\tau}{\longrightarrow} R \{y/x\} \mid (P \{y/x\} \mid Q \{y/x\}), \text{ where } y \neq x\]
\[(x)\overline{a}x.P \mid (y)ay.Q \overset{\tau}{\longrightarrow} (z)(P \{z/x\} \mid Q \{z/y\}), \text{ where } z \text{ is fresh}\]

Communications of this language are asymmetric because one has

\[\overline{a}y.P \mid (x)ax.Q \overset{\tau}{\longrightarrow} P \mid Q \{y/x\}\]

but not

\[(y)\overline{a}y.P \mid ax.Q \overset{\tau}{\longrightarrow} P \{x/y\} \mid Q\]

For more about the operational semantics, see \([7,8,9,10,11]\).

In addition to the prefix \(\alpha x\), we need some auxiliary prefixes:

\[\alpha(x).P \overset{\text{def}}{=} (x)\alpha x.P, \quad \text{bound prefix}\]
\[\langle y/x \rangle.P \overset{\text{def}}{=} (a)(\overline{a}y \mid ax.P), \quad \text{update prefix}\]
\[(y/x).P \overset{\text{def}}{=} (y)\langle y/x \rangle.P, \quad \text{bound update prefix}\]
\[\tau.P \overset{\text{def}}{=} (a)\langle a/a \rangle.P, \quad \text{tau prefix}\]

where \(x \notin \{\alpha, \overline{a}\}\), \(x \neq y\) and \(a\) is fresh. The set of all prefixes will also be ranged over by \(\lambda\).

We will write \(\phi\) and \(\psi\), called conditions, to stand for sequences of match and mismatch combinators concatenated one after another, and \(\delta\) for a sequence of mismatch operators. The notation \(\phi \Rightarrow \psi\) says that \(\phi\) logically implies \(\psi\) and \(\phi \Leftrightarrow \psi\) that \(\phi\) and \(\psi\) are logically equivalent. The concatenation of \(\psi\) and \(\phi\) is denoted by \(\psi \phi\).

A substitution \(\sigma\) respects \(\psi\) if \(\psi \Rightarrow x= y\) implies \(\sigma(x) = \sigma(y)\) and \(\psi \Rightarrow x \neq y\) implies \(\sigma(x) \neq \sigma(y)\). Dually \(\psi\) respects \(\sigma\) if \(\sigma(x) = \sigma(y)\) implies \(\psi \Rightarrow x= y\) and \(\sigma(x) \neq \sigma(y)\) implies \(\psi \Rightarrow x \neq y\). The substitution \(\sigma\) agrees with \(\psi\), and \(\psi\) agrees with \(\sigma\), if they respect each other. The substitution \(\sigma\) is induced by \(\psi\) if \(\psi\) agrees with \(\psi\) and \(n(\sigma) \subseteq n(\psi)\).

Let \(V\) be a finite set of names. We say that \(\phi\) is complete on \(V\) if \(n(\phi) = V\) and for each pair \(x, y\) of names in \(V\) it holds that either \(\phi \Rightarrow x = y\) or \(\phi \Rightarrow x \neq y\).

A sequence of names \(x_1, \ldots, x_n\) will be abbreviated as \(x\). Suppose \(Y\) is a finite set \(\{y_1, \ldots, y_n\}\) of names. The notation \([y_1 \ldots y_n]P\) will stand for \([y_1 \neq y_2] \ldots [y_1 \neq y_n]P\).

Let \(\Rightarrow\) be the reflexive and transitive closure of \(\overset{\tau}{\longrightarrow}\). We will write \(\overset{\lambda}{\Rightarrow}\) for \(\Rightarrow\). We will also write \(\overset{\lambda}{\Rightarrow}\) for \(\overset{\lambda}{\Rightarrow}\) if \(\lambda \neq \tau\) and for \(\Rightarrow\) otherwise.

In the rest of this paper we often come across with situations where we have to deal with a sequence of actions. The principal case is this:

\[P \overset{(y_1)/x(y_2)/y_1}{\Rightarrow} \ldots \overset{(y_n)/y_{n-1}}{\Rightarrow} P'\]

When \(n = 0\) we shall understand that the above sequence denotes \(P \Rightarrow P'\) and that \(y_n\) denotes \(x\).
3 Barbed Bisimulation

It has now become a routine to look at the barbed bisimilarity [13] when one investigates a new process calculus. For a channel based process calculus, the barbed approach often gives rise to the weakest bisimulation equivalence for the calculus.

**Definition 1.** A process $P$ is strongly barbed at $a$, notation $P \downarrow a$, if $P \xrightarrow{\alpha(x)} P'$ or $P \xrightarrow{\tau \subseteq} P'$ for some $P'$ such that $a \in \{\alpha, \tau\}$. $P$ is barbed at $a$, written $P \downarrow a$, if some $P'$ exists such that $P \rightarrow P' \downarrow a$. A binary relation $\mathcal{R}$ is barbed if $\forall a \in \mathcal{N}$, $P \downarrow a \Leftrightarrow Q \downarrow a$ whenever $P \mathcal{R} Q$.

The notation of context is also important to the semantic investigation. Contexts are defined inductively as follows: (i) $[]$ is a context; (ii) if $C[]$ is a context then $\alpha x.C[]$, $C[] | P$, $P | C[]$, $(x)C[]$ and $[x=y]C[]$ are contexts. Full contexts satisfy additionally: (iii) if $C[]$ is a context then $C[] + P$, $P + C[]$ and $[x \neq y]C[]$ are contexts. We are now ready to define barbed bisimulation.

**Definition 2.** Let $\mathcal{R}$ be a barbed symmetric relation closed under context. The relation $\mathcal{R}$ is a barbed bisimulation if whenever $Q \mathcal{R} P \rightarrow P'$ then $Q \rightarrow Q' \mathcal{R} P'$ for some $Q'$. The barbed bisimilarity $\approx_b$ is the largest barbed bisimulation.

In the presence of the asymmetry and the mismatch operator, the barbed bisimilarity is extremely complex. Let’s take a look at three simple examples:

- The first is concerned with (bound) updates. The process $(z/x).(y/z).Q$ is barbed bisimilar to $(z/x).(y/z).Q + [x \neq y](y/x).Q\{x/z\}$. For either process to have any effect on an environment the name $x$ must be bound, in which case both processes would have the same effect on the environment. The following internal communication for instance

  $$(x)( ((z/x).(y/z).Q + [x \neq y](y/x).Q\{x/z\}) | P) \rightarrow (Q\{x/z\}\{y/x\} | P\{y/x\})$$

is simulated by two consecutive communications

  $$(x)((z/x).(y/z).Q | P) \rightarrow (z)( (y/z).Q\{z/x\} | P\{z/x\})$$

  $$(Q\{x/z\}\{y/z\} | P\{z/x\}\{y/z\})$$

Notice that in order for the component $[x \neq y](y/x).Q\{x/z\}$ to invoke a communication it must be placed in an environment in which $x$ is localized.

- The second is about free input actions. One could argue that

  $$ax.[x \neq y]r.Q\{x/z\} + (z/x).ay.(y/z).Q + [x \neq y]ax.Q\{x/z\}$$

is barbed bisimilar to $ax.[x \neq y]r.Q\{x/z\} + (z/x).ay.(y/z).Q$. There are two main situations in which the component $[x \neq y]ax.Q\{x/z\}$ may participate in an internal communication. If $x$ is instantiated by some name different from $y$ then the component $ax.[x \neq y]r.Q\{x/z\}$ can simulate the action since the mismatch $x \neq y$ would be valid as long as $x$ is replaced by some name that is not $y$. Otherwise the component $(z/x).ay.(y/z).Q$ does the job.
The third is about bound output actions. The process

\[ \overline{a}(z).[(z \neq y](x/z).Q + (z/y).\overline{a}z.(x/z).Q + \overline{a}(x).Q\{x/z\}] \]

is barbed bisimilar to \( \overline{a}(z).[(z \neq y](x/z).Q + (z/y).\overline{a}z.(x/z).Q \). For instance

\[ (y)(ay.P | (\overline{a}(z).[(z \neq y](x/z).Q + (z/y).\overline{a}z.(x/z).Q + \overline{a}(x).Q\{x/z\})) \]

\[ \xrightarrow{\tau} (x)(P\{x/y\} | Q\{x/z\}\{x/y\}) \]

is simulated by

\[ (y)(ay.P | (\overline{a}(z).[(z \neq y](x/z).Q + (z/y).\overline{a}z.(x/z).Q)) \]

\[ \xrightarrow{\tau} (z)(az.P\{z/y\} | \overline{a}z.(x/z).Q\{z/y\}) \]

\[ \xrightarrow{\tau} (z)(P\{z/y\} | (x/z).Q\{z/y\}) \]

\[ \xrightarrow{\tau} (x)(P\{z/y\}\{x/z\} | Q\{z/y\}\{x/z\}) \]

It should be obvious from the above examples that, from the viewpoint of the barbed bisimilarity, a non-tau action of a process could be simulated by several sequences of non-tau actions of a bisimilar process. An action could be simulated by different sequences of actions in different environments. Different actions are simulated by different sequences of actions. And these sequences of actions are of different shapes.

A more interesting aspect of the barbed bisimilarity is to figure out in what manners a given action could be simulated. Let’s see an example. Suppose \( P \approx_b Q \) and \( P \xrightarrow{ax} P' \). For the barbed bisimilarity the free input action is not directly observable. What an observer can do is to observe the consequences of the action by putting \( P \) in an environment. By doing that the observer gets to know the effects of the action on the environment. Therefore in order to see how \( Q \) might simulate \( P \) we need to put them in the same environments and analyze the operational behaviours of \( Q \). There are two major cases:

- Case one:
  - Consider first the context \( _- | \overline{a}x.b(u) \) for fresh \( b, u \). Then \( P | \overline{a}x.b(u) \xrightarrow{\tau} P' | b(u) \xrightarrow{b(u)} P' | 0 \). Suppose it is simulated by \( Q | \overline{a}x.b(u) \xrightarrow{b(u)} Q' | 0 \approx_b P' | 0 \). This sequence of actions can be factorized in two manners: Either

    \[ Q | \overline{a}x.b(u) \Rightarrow Q_1 | \overline{a}x.b(u) \xrightarrow{\tau} Q_2 | b(u) \xrightarrow{b(u)} Q_2 | 0 \Rightarrow Q' | 0 \]

    or

    \[ Q | \overline{a}x.b(u) \Rightarrow Q_1 | \overline{a}x.b(u) \xrightarrow{\tau} Q''(x/z) | b(u) \xrightarrow{b(u)} Q''(x/z) | 0 \Rightarrow Q' | 0 \]

  - In the first case, \( Q \xrightarrow{ax} Q' \approx_b P' \). In the second case, \( Q \xrightarrow{ax} Q'' \) and \( Q''(x/z) \Rightarrow Q' \approx_b P' \).
• Consider now the context $(x)(\_ | \overline{\alpha} z. \overline{b} x)$ for fresh $b, z$. Then

$$(x)(P \mid \overline{\alpha} z. \overline{b} x) \xrightarrow{\tau} P' \{z/x\} \mid \overline{b} z \xrightarrow{\overline{b} z} P' \{z/x\} \mid 0$$

This sequence of actions can be matched up by $(x)(Q \mid \overline{\alpha} z. \overline{b} x) \xrightarrow{\overline{b} z} Q' \mid 0$, which can be factorized in two ways: Either

$$(x)(Q \mid \overline{\alpha} z. \overline{b} x) \Longrightarrow (y_n)(Q_1 \mid \overline{\alpha} z. \overline{b} y_n)$$

\[ \xrightarrow{\tau} Q_2 \mid \overline{b} z \]

\[ \xrightarrow{\overline{b} z} Q_2 \mid 0 \]

\[ \Longrightarrow Q' \mid 0 \]

or

$$(x)(Q \mid \overline{\alpha} z. \overline{b} x) \Longrightarrow (y_n)(Q_1 \mid \overline{\alpha} z. \overline{b} y_n)$$

\[ \Longrightarrow (y_n)(Q_2 \mid \overline{b} y_n) \]

\[ \Longrightarrow Q_3 \mid \overline{b} z \]

\[ \xrightarrow{\overline{b} z} Q_3 \mid 0 \]

\[ \Longrightarrow Q' \mid 0 \]

In the former case

$$Q \xrightarrow{(y_1)/x(y_2)/y_1 \ldots (y_n)/y_{n-1} a y_n} Q_2 \{y_n/z\} \Longrightarrow Q' \{y_n/z\}$$

and $Q' \{y_n/z\} \{x/y_n\} \ni_b P'$ for some $y_1, \ldots, y_n$, where $n \geq 0$. In the latter case

$$Q \xrightarrow{(y_1)/x(y_2)/y_1 \ldots (y_n)/y_{n-1} a(z_1)/y_n(z_2)/z_1 \ldots (z_m)/y_{n-1} z_2/z_m} Q'$$

and $Q' \{x/z\} \ni_b P'$ for some $z, y_1, \ldots, y_n, z_1, \ldots, z_m$, where $n \geq 0$ and $m \geq 0$.

Case two: For each $y$ distinct from $x$ one has

$$(x)(P \mid \overline{\alpha} y.[x=y] b(u)) \xrightarrow{\tau} P' \{y/x\} \mid [y=y] b(u) \xrightarrow{b(u)} P' \{y/x\} \mid 0$$

for fresh names $b$ and $u$. It follows from $P \approx_b Q$ that $(x)(Q \mid \overline{\alpha} y.[x=y] b(u)) \xrightarrow{b(u)} Q' \mid 0 \approx_b P' \{y/x\} \mid 0$, which can be factorized in five different fashions:

• The first factorization is

$$(x)(Q \mid \overline{\alpha} y.[x=y] b(u)) \Longrightarrow (y_n)(Q_1 \mid \overline{\alpha} y.[y_n=y] b(u))$$

\[ \xrightarrow{\tau} Q_2 \{y/y_n\} \mid [y=y] b(u) \]

\[ \xrightarrow{b(u)} Q_2 \{y/y_n\} \mid 0 \]

\[ \Longrightarrow Q' \mid 0 \]

Obviously