Mario Coppo
Elena Lodi
G. Michele Pinna (Eds.)

LNCS 3701

Theoretical Computer Science

9th Italian Conference, ICTCS 2005
Siena, Italy, October 2005
Proceedings

Springer

Mario Coppo Elena Lodi
G. Michele Pinna (Eds.)

Theoretical Computer Science

9th Italian Conference, ICTCS 2005
Siena, Italy, October 12-14, 2005
Proceedings

Springer

Volume Editors

Mario Coppo
Università di Torino, Dipartimento di Informatica
C. Svizzera 185, 10149 Torino, Italy
E-mail: coppo@di.unito.it

Elena Lodi
Università di Siena
Dipartimento di Scienze Matematiche e Informatiche
Pian dei Mantellini 44, 53100 Siena, Italy
E-mail: lodi@unisi.it

G. Michele Pinna
Università di Siena
Dipartimento di Scienze Matematiche e Informatiche
Pian dei Mantellini 44, 53100 Siena, Italy
and Università di Cagliari
Dipartimento di Informatica e Matematica
Via Ospedale 72, 09124 Cagliari, Italy
E-mail: gmpinna@unica.it

Library of Congress Control Number: 2005933156

CR Subject Classification (1998): F, E.1, G.1-2

ISSN 0302-9743
ISBN-10 3-540-29106-7 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-29106-0 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11560586 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3701

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Preface

The 9th Italian Conference on Theoretical Computer Science (ICTCS 2005) was held at the Certosa di Pontignano, Siena, Italy, on October 12–14 2005. The Certosa di Pontignano is the conference center of the University of Siena; it is located 8 km away from the town and it is in the Chianti region. The Certosa is a place full of history (founded in the 15th century, it was set on fire a century later and reconstructed) and of valuable artworks, like frescoes of the Scuola Senese.

Previous conferences took place in Pisa (1972), Mantova (1974 and 1989), L'Aquila (1992), Ravello (1995), Prato (1998), Turin (2001) and Bertinoro (2003).

The conference aims at bringing together computer scientists, especially young researchers, to foster cooperation, exchange of ideas and results. Great efforts have been made to attract researchers from all over the world. The main topics of the conference cover all the fields of theoretical computer science and include analysis and design of algorithms, computability, computational complexity, cryptography, formal languages and automata, foundations of programming languages and program analysis, foundations of artificial intelligence and knowledge representation, foundations of web programming, natural computing paradigms (quantum computing, bioinformatics), parallel and distributed computation, program specification and verification, term rewriting, theory of concurrency, theory of data bases, theory of logical design and layout, type theory, security, and symbolic and algebraic computation.

The Program Committee, consisting of 16 members, considered 83 papers and selected 29 for presentation. These papers were selected, after a careful reviewing process, on the basis of their originality, quality and relevance to theoretical computer science. In keeping with the previous conferences, ICTCS 2005 was characterized by the high number of submissions and by the number of different countries represented. These proceedings contain the revised versions of the 29 accepted papers together with the invited talks by Luca Cardelli (Biological Systems as Reactive Systems, abstract), Giuseppe Castagna (Semantic Subtyping: Challenges, Perspectives, and Open Problems) and Nicola Santoro (Mobile Agents Computing: Security Issues and Algorithmic Solutions).

Due to the high quality of the submissions, paper selection was a difficult and challenging task. Each submission was reviewed by at least three reviewers. We thank all the Program Committee members and the additional reviewers for their accurate work and for spending so much time in the reviewing process. We apologize for any inadvertent omission in the list of reviewers.

Following the example of the last ICTCS edition, we encouraged authors to submit their papers in electronic format. Special thanks are due to Simone Donetti for setting up a very friendly Web site for this purpose.

Finally, we would like to thank all the authors who submitted papers and all the conference participants.

October 2005

Mario Coppo
Elena Lodi
G. Michele Pinna

ICTCS 2005

October 12–14, Certosa di Pontignano, Siena, Italy

Program Co-chairs

Mario Coppo	Università di Torino
Elena Lodi	Università di Siena

Program Committee

Michele Bugliesi	Università di Venezia
Mario Coppo	Università di Torino (Co-chair)
Pierluigi Crescenzi	Università di Firenze
Giulia Galbiati	Università di Pavia
Luisa Gargano	Università di Salerno
Giorgio Ghelli	Università di Pisa
Roberto Grossi	Università di Pisa
Benedetto Intrigila	Università di L'Aquila
Nicola Leone	Università di Cosenza
Elena Lodi	Università di Siena, (Co-chair)
Flaminia Luccio	Università di Trieste
Andrea Masini	Università di Verona
Giancarlo Mauri	Università di Milano-Bicocca
Corrado Priami	Università di Trento
Geppino Pucci	Università di Padova
Davide Sangiorgi	Università di Bologna

Organizing Committee

G. Michele Pinna	Università di Cagliari (Chair)
Sara Brunetti	Università di Siena
Elisa Tiezzi	Università di Siena

Sponsoring Institutions

European Association for Theoretical Computer Science (EATCS)
Dipartimento di Matematica e Informatica "R. Magari", Università di Siena
Dipartimento di Informatica, Università di Torino
Università degli Studi di Siena
Monte dei Paschi di Siena

Referees

Tetsuo Asano
Vincenzo Auletta
Benjamin Aziz
Paolo Baldan
Anindya Banerjee
Stefano Baratella
Franco Barbanera
Massimo Bartoletti
Marie-Pierre Béal
Stefano Berardi
Marco Bernardo
Alberto Bertoni
Daniela Besozzi
Chiara Bodei
Paolo Boldi
Hanifa Boucheneb
Linda Brodo
Sara Brunetti
Francesco Buccafurri
Pasquale Caianiello
Tiziana Calamoneri
Felice Cardone
Dario Catalano
Marco Cesati
Federica Ciocchetta
Valentina Ciriani
Antonio Cisternino
Dario Colazzo
Andrea Clementi
Carlo Combi
Paolo D'Arco
Ottavio D'Antona
Christophe Decanniere
Pierpaolo Degano
Giuseppe Della Penna
Gianluca Della Vedova
Gianluca De Marco
Maria Rita Di Berardini
Pietro Di Giannantonio
Susanna Donatelli
Riccardo Dondi
Claudio Eccher
Wolfgang Faber
Riccardo Focardi
Paola Flocchini
Andrea Frosini
Clemente Galdi
Dora Giammarresi
Paola Giannini
Laura Giordano
Gianluigi Greco
Massimiliano Goldwurm
Stefano Guerrini
Giovambattista Ianni
Amos Korman
Ruggero Lanotte
Paola Lecca
Alberto Leporati
Renato Locigno
Fabrizio Luccio
Veli Mäkinen
Alessio Malizia
Stefano Mancini
Alberto Marchetti Spaccamela
Radu Mardare
Luciano Margara
Ines Margaria
Carlo Mereghetti
Alberto Martelli
Donatella Merlini
Filippo Mignosi
Alberto Momigliano
Manuela Montangero
Elisa Mori
Ian Munro
Aniello Murano
Venkatesh Mysore
Matthias Neubauer
Monica Nesi
Mitsunori Ogihara
Linda Pagli
Beatrice Palano
Luigi Palopoli
David Peleg
Paolo Penna
Simona Perri
Carla Piazza
Adolfo Piperno
Nadia Pisanti
Katerina Pokozy
Roberto Posenato
Davide Prandi
Giuseppe Prencipe
Orazio Puglisi
Paola Quaglia
Sven Rahmann
Adele Rescigno
Antonio Restivo
Simone Rinaldi
Lorenzo Robbiano
Simona Ronchi della Rocca
Gianluca Rossi
Alehandro Russo
Giancarlo Ruffo
Ivano Salvo
Francesco Scarcello
Debora Schuch
Roberto Segala
Andrea Sgarro
Riccardo Silvestri
Maria Simi
Robert Spalek
Renzo Sprugnoli
Frank Christian Stephan
Giorgio Terracina
Elisa Tiezzi
Mauro Torelli
Andrea Torsello
Emilio Tuosto
Ugo Vaccaro
Leonardo Vanneschi
Stefano Varricchio
Maria Cecilia Verri
Bob Walters
Damiano Zanardini
Claudio Zandron
Nicola Zannone

Lecture Notes in Computer Science

For information about Vols. 1–3861

please contact your bookseller or Springer

Vol. 3987: M. Hazas, J. Krumm, T. Strang (Eds.), Location- and Context-Awareness. X, 289 pages. 2006.

Vol. 3968: K.P. Fishkin, B. Schiele, P. Nixon, A. Quigley (Eds.), Pervasive Computing. XV, 402 pages. 2006.

Vol. 3959: J.-Y. Cai, S. B. Cooper, A. Li (Eds.), Theory and Applications of Models of Computation. XV, 794 pages. 2006.

Vol. 3958: M. Yung, Y. Dodis, A. Kiayias, T. Malkin (Eds.), Public Key Cryptography - PKC 2006. XIV, 543 pages. 2006.

Vol. 3956: G. Barthe, B. Gregoire, M. Huisman, J.-L. Lanet (Eds.), Construction and Analysis of Safe, Secure, and Interoperable Smart Devices. IX, 175 pages. 2006.

Vol. 3955: G. Antoniou, G. Potamias, C. Spyropoulos, D. Plexousakis (Eds.), Advances in Artificial Intelligence. XVII, 611 pages. 2006. (Sublibrary LNAI).

Vol. 3954: A. Leonardis, H. Bischof, A. Pinz (Eds.), Computer Vision – ECCV 2006, Part IV. XVII, 613 pages. 2006.

Vol. 3953: A. Leonardis, H. Bischof, A. Pinz (Eds.), Computer Vision – ECCV 2006, Part III. XVII, 649 pages. 2006.

Vol. 3952: A. Leonardis, H. Bischof, A. Pinz (Eds.), Computer Vision – ECCV 2006, Part II. XVII, 661 pages. 2006.

Vol. 3951: A. Leonardis, H. Bischof, A. Pinz (Eds.), Computer Vision – ECCV 2006, Part I. XXXV, 639 pages. 2006.

Vol. 3947: Y.-C. Chung, J.E. Moreira (Eds.), Advances in Grid and Pervasive Computing. XXI, 667 pages. 2006.

Vol. 3946: T.R. Roth-Berghofer, S. Schulz, D.B. Leake (Eds.), Modeling and Retrieval of Context. XI, 149 pages. 2006. (Sublibrary LNAI).

Vol. 3945: M. Hagiya, P. Wadler (Eds.), Functional and Logic Programming. X, 295 pages. 2006.

Vol. 3944: J. Quiñonero-Candela, I. Dagan, B. Magnini, F. d'Alché-Buc (Eds.), Machine Learning Challenges. XIII, 462 pages. 2006. (Sublibrary LNAI).

Vol. 3942: Z. Pan, R. Aylett, H. Diener, X. Jin, S. Göbel, L. Li (Eds.), Technologies for E-Learning and Digital Entertainment. XXV, 1396 pages. 2006.

Vol. 3939: C. Priami, L. Cardelli, S. Emmott (Eds.), Transactions on Computational Systems Biology IV. VII, 141 pages. 2006. (Sublibrary LNBI).

Vol. 3936: M. Lalmas, A. MacFarlane, S. Rüger, A. Tombros, T. Tsikrika, A. Yavlinsky (Eds.), Advances in Information Retrieval. XIX, 584 pages. 2006.

Vol. 3935: D. Won, S. Kim (Eds.), Information Security and Cryptology - ICISC 2005. XIV, 458 pages. 2006.

Vol. 3934: J.A. Clark, R.F. Paige, F.A. C. Polack, P.J. Brooke (Eds.), Security in Pervasive Computing. X, 243 pages. 2006.

Vol. 3933: F. Bonchi, J.-F. Boulicaut (Eds.), Knowledge Discovery in Inductive Databases. VIII, 251 pages. 2006.

Vol. 3931: B. Apolloni, M. Marinaro, G. Nicosia, R. Tagliaferri (Eds.), Neural Nets. XIII, 370 pages. 2006.

Vol. 3930: D.S. Yeung, Z.-Q. Liu, X.-Z. Wang, H. Yan (Eds.), Advances in Machine Learning and Cybernetics. XXI, 1110 pages. 2006. (Sublibrary LNAI).

Vol. 3929: W. MacCaull, M. Winter, I. Düntsch (Eds.), Relational Methods in Computer Science. VIII, 263 pages. 2006.

Vol. 3928: J. Domingo-Ferrer, J. Posegga, D. Schreckling (Eds.), Smart Card Research and Advanced Applications. XI, 359 pages. 2006.

Vol. 3927: J. Hespanha, A. Tiwari (Eds.), Hybrid Systems: Computation and Control. XII, 584 pages. 2006.

Vol. 3925: A. Valmari (Ed.), Model Checking Software. X, 307 pages. 2006.

Vol. 3924: P. Sestoft (Ed.), Programming Languages and Systems. XII, 343 pages. 2006.

Vol. 3923: A. Mycroft, A. Zeller (Eds.), Compiler Construction. XIII, 277 pages. 2006.

Vol. 3922: L. Baresi, R. Heckel (Eds.), Fundamental Approaches to Software Engineering. XIII, 427 pages. 2006.

Vol. 3921: L. Aceto, A. Ingólfsdóttir (Eds.), Foundations of Software Science and Computation Structures. XV, 447 pages. 2006.

Vol. 3920: H. Hermanns, J. Palsberg (Eds.), Tools and Algorithms for the Construction and Analysis of Systems. XIV, 506 pages. 2006.

Vol. 3918: W.K. Ng, M. Kitsuregawa, J. Li, K. Chang (Eds.), Advances in Knowledge Discovery and Data Mining. XXIV, 879 pages. 2006. (Sublibrary LNAI).

Vol. 3917: H. Chen, F.Y. Wang, C.C. Yang, D. Zeng, M. Chau, K. Chang (Eds.), Intelligence and Security Informatics. XII, 186 pages. 2006.

Vol. 3916: J. Li, Q. Yang, A.-H. Tan (Eds.), Data Mining for Biomedical Applications. VIII, 155 pages. 2006. (Sublibrary LNBI).

Vol. 3915: R. Nayak, M.J. Zaki (Eds.), Knowledge Discovery from XML Documents. VIII, 105 pages. 2006.

Vol. 3914: A. Garcia, R. Choren, C. Lucena, P. Giorgini, T. Holvoet, A. Romanovsky (Eds.), Software Engineering for Multi-Agent Systems IV. XIV, 255 pages. 2006.

Vol. 3910: S.A. Brueckner, G.D.M. Serugendo, D. Hales, F. Zambonelli (Eds.), Engineering Self-Organising Systems. XII, 245 pages. 2006. (Sublibrary LNAI).

Vol. 3909: A. Apostolico, C. Guerra, S. Istrail, P. Pevzner, M. Waterman (Eds.), Research in Computational Molecular Biology. XVII, 612 pages. 2006. (Sublibrary LNBI).

Vol. 3908: A. Bui, M. Bui, T. Böhme, H. Unger (Eds.), Innovative Internet Community Systems. VIII, 207 pages. 2006.

Vol. 3907: F. Rothlauf, J. Branke, S. Cagnoni, E. Costa, C. Cotta, R. Drechsler, E. Lutton, P. Machado, J.H. Moore, J. Romero, G.D. Smith, G. Squillero, H. Takagi (Eds.), Applications of Evolutionary Computing. XXIV, 813 pages. 2006.

Vol. 3906: J. Gottlieb, G.R. Raidl (Eds.), Evolutionary Computation in Combinatorial Optimization. XI, 293 pages. 2006.

Vol. 3905: P. Collet, M. Tomassini, M. Ebner, S. Gustafson, A. Ekárt (Eds.), Genetic Programming. XI, 361 pages. 2006.

Vol. 3904: M. Baldoni, U. Endriss, A. Omicini, P. Torroni (Eds.), Declarative Agent Languages and Technologies III. XII, 245 pages. 2006. (Sublibrary LNAI).

Vol. 3903: K. Chen, R. Deng, X. Lai, J. Zhou (Eds.), Information Security Practice and Experience. XIV, 392 pages. 2006.

Vol. 3901: P.M. Hill (Ed.), Logic Based Program Synthesis and Transformation. X, 179 pages. 2006.

Vol. 3900: F. Toni, P. Torroni (Eds.), Computational Logic in Multi-Agent Systems. XVII, 427 pages. 2006. (Sublibrary LNAI).

Vol. 3899: S. Frintrop, VOCUS: A Visual Attention System for Object Detection and Goal-Directed Search. XIV, 216 pages. 2006. (Sublibrary LNAI).

Vol. 3898: K. Tuyls, P.J. 't Hoen, K. Verbeeck, S. Sen (Eds.), Learning and Adaption in Multi-Agent Systems. X, 217 pages. 2006. (Sublibrary LNAI).

Vol. 3897: B. Preneel, S. Tavares (Eds.), Selected Areas in Cryptography. XI, 371 pages. 2006.

Vol. 3896: Y. Ioannidis, M.H. Scholl, J.W. Schmidt, F. Matthes, M. Hatzopoulos, K. Boehm, A. Kemper, T. Grust, C. Boehm (Eds.), Advances in Database Technology - EDBT 2006. XIV, 1208 pages. 2006.

Vol. 3895: O. Goldreich, A.L. Rosenberg, A.L. Selman (Eds.), Theoretical Computer Science. XII, 399 pages. 2006.

Vol. 3894: W. Grass, B. Sick, K. Waldschmidt (Eds.), Architecture of Computing Systems - ARCS 2006. XII, 496 pages. 2006.

Vol. 3893: L. Atzori, D.D. Giusto, R. Leonardi, F. Pereira (Eds.), Visual Content Processing and Representation. IX, 224 pages. 2006.

Vol. 3891: J.S. Sichman, L. Antunes (Eds.), Multi-Agent-Based Simulation VI. X, 191 pages. 2006. (Sublibrary LNAI).

Vol. 3890: S.G. Thompson, R. Ghanea-Hercock (Eds.), Defence Applications of Multi-Agent Systems. XII, 141 pages. 2006. (Sublibrary LNAI).

Vol. 3889: J. Rosca, D. Erdogmus, J.C. Príncipe, S. Haykin (Eds.), Independent Component Analysis and Blind Signal Separation. XXI, 980 pages. 2006.

Vol. 3888: D. Draheim, G. Weber (Eds.), Trends in Enterprise Application Architecture. IX, 145 pages. 2006.

Vol. 3887: J.R. Correa, A. Hevia, M. Kiwi (Eds.), LATIN 2006: Theoretical Informatics. XVI, 814 pages. 2006.

Vol. 3886: E.G. Bremer, J. Hakenberg, E.-H.(S.) Han, D. Berrar, W. Dubitzky (Eds.), Knowledge Discovery in Life Science Literature. XIV, 147 pages. 2006. (Sublibrary LNBI).

Vol. 3885: V. Torra, Y. Narukawa, A. Valls, J. Domingo-Ferrer (Eds.), Modeling Decisions for Artificial Intelligence. XII, 374 pages. 2006. (Sublibrary LNAI).

Vol. 3884: B. Durand, W. Thomas (Eds.), STACS 2006. XIV, 714 pages. 2006.

Vol. 3882: M.L. Lee, K.-L. Tan, V. Wuwongse (Eds.), Database Systems for Advanced Applications. XIX, 923 pages. 2006.

Vol. 3881: S. Gibet, N. Courty, J.-F. Kamp (Eds.), Gesture in Human-Computer Interaction and Simulation. XIII, 344 pages. 2006. (Sublibrary LNAI).

Vol. 3880: A. Rashid, M. Aksit (Eds.), Transactions on Aspect-Oriented Software Development I. IX, 335 pages. 2006.

Vol. 3879: T. Erlebach, G. Persinao (Eds.), Approximation and Online Algorithms. X, 349 pages. 2006.

Vol. 3878: A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing. XVII, 589 pages. 2006.

Vol. 3877: M. Detyniecki, J.M. Jose, A. Nürnberger, C. J. '. van Rijsbergen (Eds.), Adaptive Multimedia Retrieval: User, Context, and Feedback. XI, 279 pages. 2006.

Vol. 3876: S. Halevi, T. Rabin (Eds.), Theory of Cryptography. XI, 617 pages. 2006.

Vol. 3875: S. Ur, E. Bin, Y. Wolfsthal (Eds.), Hardware and Software, Verification and Testing. X, 265 pages. 2006.

Vol. 3874: R. Missaoui, J. Schmidt (Eds.), Formal Concept Analysis. X, 309 pages. 2006. (Sublibrary LNAI).

Vol. 3873: L. Maicher, J. Park (Eds.), Charting the Topic Maps Research and Applications Landscape. VIII, 281 pages. 2006. (Sublibrary LNAI).

Vol. 3872: H. Bunke, A. L. Spitz (Eds.), Document Analysis Systems VII. XIII, 630 pages. 2006.

Vol. 3871: E.-G. Talbi, P. Liardet, P. Collet, E. Lutton, M. Schoenauer (Eds.), Artificial Evolution. XI, 310 pages. 2006.

Vol. 3870: S. Spaccapietra, P. Atzeni, W.W. Chu, T. Catarci, K.P. Sycara (Eds.), Journal on Data Semantics V. XIII, 237 pages. 2006.

Vol. 3869: S. Renals, S. Bengio (Eds.), Machine Learning for Multimodal Interaction. XIII, 490 pages. 2006.

Vol. 3868: K. Römer, H. Karl, F. Mattern (Eds.), Wireless Sensor Networks. XI, 342 pages. 2006.

Vol. 3866: T. Dimitrakos, F. Martinelli, P.Y.A. Ryan, S. Schneider (Eds.), Formal Aspects in Security and Trust. X, 259 pages. 2006.

Vol. 3865: W. Shen, K.-M. Chao, Z. Lin, J.-P.A. Barthès, A. James (Eds.), Computer Supported Cooperative Work in Design II. XII, 659 pages. 2006.

Vol. 3863: M. Kohlhase (Ed.), Mathematical Knowledge Management. XI, 405 pages. 2006. (Sublibrary LNAI).

Vol. 3862: R.H. Bordini, M. Dastani, J. Dix, A.E.F. Seghrouchni (Eds.), Programming Multi-Agent Systems. XIV, 267 pages. 2006. (Sublibrary LNAI).

Table of Contents

Invited Contributions

Technical Contributions

Semantic Subtyping: Challenges, Perspectives, and Open Problems

Giuseppe Castagna

CNRS, École Normale Supérieure de Paris, France

Based on joint work with: Véronique Benzaken, Rocco De Nicola, Mariangiola Dezani, Alain Frisch, Haruo Hosoya, Daniele Varacca

Abstract. Semantic subtyping is a relatively new approach to define subtyping relations where types are interpreted as sets and union, intersection and negation types have the corresponding set-theoretic interpretation. In this lecture we outline the approach, give an aperçu of its expressiveness and generality by applying it to the λ-calculus with recursive and product types and to the π-calculus. We then discuss in detail the new challenges and research perspectives that the approach brings forth.

1 Introduction to the Semantic Subtyping

Many recent type systems rely on a subtyping relation. Its definition generally depends on the type algebra, and on its intended use. We can distinguish two main approaches for defining subtyping: the *syntactic* approach and the *semantic* one. The syntactic approach—by far the more used—consists in defining the subtyping relation by axiomatising it in a formal system (a set of inductive or coinductive rules); in the semantic approach (for instance, [AW93, Dam94]), instead, one starts with a model of the language and an interpretation of types as subsets of the model, then defines the subtyping relation as the inclusion of denoted sets, and, finally, when the relation is decidable, derives a subtyping algorithm from the semantic definition.

The semantic approach has several advantages (see [CF05] for an overview) but it is also more constraining. Finding an interpretation in which types can be interpreted as subsets of a model may be a hard task. A solution to this problem was given by Haruo Hosoya and Benjamin Pierce [HP01, Hos01, HP03], who noticed that in order to define subtyping all is needed is a set theoretic interpretation of types, not a model of the terms. In particular, they propose to interpret a type as the set of all values that have that type. So if we use $\mathcal{V}$ to denote the set of all values, then we can define the following set-theoretic interpretation for types $[\![t]\!]_{\mathcal{V}} = \{v \in \mathcal{V} \mid \vdash v : t\}$ which induces the following subtyping relation:

$$s \leq_{\mathcal{V}} t \quad \stackrel{def}{\iff} \quad [\![s]\!]_{\mathcal{V}} \subseteq [\![t]\!]_{\mathcal{V}} \tag{1}$$

This works for Hosoya and Pierce because the set of values they consider can be defined independently from the typing relation.[1] But in general in order to state when a value has a given type (the "$\vdash v : t$" in the previous definition) one needs the subtyping

[1] Their values are XML documents, and they can be defined as regular trees. The typing relation, then, becomes recognition of a regular tree language.

M. Coppo et al. (Eds.): ICTCS 2005, LNCS 3701, pp. 1–20, 2005.

relation. This yields a circularity: we are building a model to define the subtyping relation, and the definition of this model needs the subtyping relation. This circularity is patent in both the examples we discuss below: in λ-calculus (Section 2) values are λ-abstractions and to type them (in particular, to type applications that may occur in their body) subtyping is needed; in π-calculus (Section 3) the covariance and contravariance of read-only and write-only channel types make the subtyping relation necessary to type channels.

In order to avoid this circularity and still interpret types as set of values, we resort to a bootstrapping technique. The general ideas of this technique are informally exposed in [CF05], while the technical development can be found in [FCB02, Fri04]. For the aims of this article, the process of defining semantic subtyping can be roughly summarised in the following steps:

1. Take a bunch of type *constructors* (e.g., $\to$, $\times$, $ch(\,)$, ...) and extend the type algebra with the following *boolean combinators*: union $\vee$, intersection $\wedge$, and negation $\neg$.
2. Give a *set-theoretic model* of the type algebra, namely define a function $[\![\,]\!]_{\mathcal{D}}$: **Types** $\to \mathcal{P}(\mathcal{D})$, for some domain $\mathcal{D}$ (where $\mathcal{P}(\mathcal{D})$ denotes the powerset of $\mathcal{D}$). In such a model, the combinators must be interpreted in a set-theoretic way (that is, $[\![s \wedge t]\!]_{\mathcal{D}} = [\![s]\!]_{\mathcal{D}} \cap [\![t]\!]_{\mathcal{D}}$, $[\![s \vee t]\!]_{\mathcal{D}} = [\![s]\!]_{\mathcal{D}} \cup [\![t]\!]_{\mathcal{D}}$, and $[\![\neg t]\!]_{\mathcal{D}} = \mathcal{D} \setminus [\![t]\!]_{\mathcal{D}}$), and the definition of the model must capture the essence of the type constructors.

   There might be several models, and each of them induces a specific subtyping relation on the type algebra. We only need to prove that there exists at least one model and then pick one that we call the *bootstrap model*. If its associated interpretation function is $[\![\,]\!]_{\mathcal{B}}$, then it induces the following subtyping relation:

$$s \leq_{\mathcal{B}} t \quad \stackrel{def}{\iff} \quad [\![s]\!]_{\mathcal{B}} \subseteq [\![t]\!]_{\mathcal{B}} \tag{2}$$

3. Now that we defined a subtyping relation for our types, find a subtyping algorithm that decides (or semi-decides) the relation. This step is not mandatory but highly advisable if we want to use our types in practise.
4. Now that we have a (hopefully) suitable subtyping relation available, we can focus on the language itself, consider its typing rules, use the new subtyping relation to type the terms of the language, and deduce $\Gamma \vdash_{\mathcal{B}} e : t$. In particular this means to use in the subsumptionrule the bootstrap subtyping relation $\leq_{\mathcal{B}}$ we defined in step 2.
5. The typing judgement for the language now allows us to define a *new* natural set-theoretic interpretation of types, the one based on values $[\![t]\!]_{\mathcal{V}} = \{v \in \mathcal{V} \mid \vdash_{\mathcal{B}} v : t\}$, and then define a "new" subtyping relation as in equation (1). This relation might be different from $\leq_{\mathcal{B}}$ we started from. However, if the definitions of the model, of the language, and of the typing rules have been carefully chosen, then the two subtyping relations coincide

$$s \leq_{\mathcal{B}} t \quad \iff \quad s \leq_{\mathcal{V}} t$$

   and this closes the circularity. Then, the rest of the story is standard (reduction relation, subject reduction, type-checking algorithm, etc ...).

The accomplishment of this is process is far from being straightforward. In point 2 it may be quite difficult to capture the semantics of the type constructors (e.g., it is quite

hard to define a set-theoretic semantics for arrow types); in point 3 defining a model may go from tricky to impossible (e.g., because of bizarre interactions with recursive types); point 4 may fail for the inability of devising a subtyping algorithm (cf. the subtyping algorithm for $\mathbb{C}\pi$ in [CNV05]); finally the last step is the most critical one since it may require a consistent rewriting of the language and/or of the typing rules to "close the circle" ... if possible at all. We will give examples of all these problems in the rest of this document.

In the next two sections we are going to show how to apply this process to λ-like and π-like calculi. The presentation will be sketchy and presuppose from the reader some knowledge of the λ-calculus, of the π-calculus, and of their type systems. Also, the calculi we are going to present are very simplified versions of the actual ones whose detailed descriptions can be found in [FCB02] and [CNV05], respectively.

2 Semantic λ-Calculus: $\mathbb{C}$Duce

As a first example of application of the semantic subtyping 5-steps technique, let us take the λ-calculus with products.

Step 1. The first step consists in taking some type constructors, in this case products and arrows, and adding boolean combinators to them:

$$t ::= \mathbb{0} \mid \mathbb{1} \mid t \to t \mid t \times t \mid \neg t \mid t \vee t \mid t \wedge t$$

where $\mathbb{0}$ and $\mathbb{1}$ correspond, respectively, to the empty and the universal types. For more generality we consider also recursive types. Thus, our types are the regular trees generated by the grammar above and satisfying the standard contractivity condition that every infinite branch has infinitely many occurrences of the $\times$ or of the $\to$ constructors (this rules out meaningless expressions such as $t \wedge (t \wedge (t \wedge (\ldots)))$).

Step 2. The second step is, in this case, the hard one as it requires to define a set-theoretic interpretation $[\![\,]\!]_{\mathcal{D}} : \mathbf{Types} \to \mathcal{P}(\mathcal{D})$. But, how can we give a set theoretic interpretation to the arrow type? The set theoretic intuition we have of $t \to s$ is that it is the set of all functions (of our language) that when applied to a value of type t either diverge or return a result of type s. If we interpret functions as binary relations on $\mathcal{D}$, then $[\![t \to s]\!]$ is the set of binary relations in which if the first projection is in (the interpretation of) t then the second projection is in (the interpretation of) s, namely $\mathcal{P}(\overline{[\![t]\!] \times \overline{[\![s]\!]}})$, where the overline denotes set complement.[2] However, setting $[\![t \to s]\!] = \mathcal{P}(\overline{[\![t]\!] \times \overline{[\![s]\!]}})$ is impossible since, for cardinality reasons, we cannot have $\mathcal{P}(\mathcal{D}^2) \subseteq \mathcal{D}$. Note though, that we do not define the interpretation $[\![\,]\!]$ in order to formally state what the syntactic types *mean* but, more simply, we define it in order to state how they are *related*. Therefore, even if the interpretation does not capture the intended semantics of types, all we need is that it captures the containment relation induced by this semantics. That is, roughly, it suffices to our aims that the interpretation function satisfies

$$[\![t_1 \to s_1]\!] \subseteq [\![t_2 \to s_2]\!] \iff \mathcal{P}(\overline{[\![t_1]\!] \times \overline{[\![s_1]\!]}}) \subseteq \mathcal{P}(\overline{[\![t_2]\!] \times \overline{[\![s_2]\!]}}) \tag{3}$$

[2] This is just one of the possible interpretations. See [CF05] for a discussion about the implications of such a choice and [Fri04] for examples of different interpretations.

Note that $\mathcal{P}_f(X) \subseteq \mathcal{P}_f(Y)$ if and only if $\mathcal{P}(X) \subseteq \mathcal{P}(Y)$ (where $\mathcal{P}_f$ denotes the finite powerset). Therefore, if we set $[\![t \to s]\!] = \mathcal{P}_f(\overline{[\![t]\!] \times \overline{[\![s]\!]}})$, this interpretation satisfies (3). In other words, we can use as bootstrap model $\mathcal{B}$ the least solution of the equation $X = X^2 + \mathcal{P}_f(X^2)$ and the following interpretation function[3] $[\![\ ]\!]_{\mathcal{B}}$: **Types** $\to \mathcal{P}(\mathcal{B})$:

$$[\![\mathbb{0}]\!]_{\mathcal{B}} = \varnothing \quad [\![\mathbb{1}]\!]_{\mathcal{B}} = \mathcal{B} \quad [\![s \vee t]\!]_{\mathcal{B}} = [\![s]\!]_{\mathcal{B}} \cup [\![t]\!]_{\mathcal{B}} \quad [\![s \wedge t]\!]_{\mathcal{B}} = [\![s]\!]_{\mathcal{B}} \cap [\![t]\!]_{\mathcal{B}}$$
$$[\![\neg t]\!]_{\mathcal{B}} = \mathcal{B} \setminus [\![t]\!]_{\mathcal{B}} \quad [\![s \times t]\!]_{\mathcal{B}} = [\![s]\!] \times [\![t]\!] \quad [\![t \to s]\!]_{\mathcal{B}} = \mathcal{P}_f(\overline{[\![t]\!]_{\mathcal{B}} \times \overline{[\![s]\!]_{\mathcal{B}}}})$$

The model we have chosen can represent only finite graph functions, therefore it is not rich enough to give semantics to a λ-calculus (even the simply typed one). However since this model satisfies equation (3), it is able to express the containment relation induced by the semantic intuition we have of the type $t \to s$ (namely that it represents $\mathcal{P}(\overline{[\![t]\!] \times \overline{[\![s]\!]}})$, which is all we need.

Step 3. We can use the definition of subtyping as given by equation (2) to deduce some interesting relations: for instance, according to (2) the type $(t_1 \to s_1) \wedge (t_2 \to s_2)$ is a subtype of $(t_1 \wedge t_2) \to (s_1 \wedge s_2)$, of $(t_1 \vee t_2) \to (s_1 \vee s_2)$, of their intersection and, in general, all these inclusions are strict.

Apart from these examples, the point of course is to devise an algorithm to decide inclusion between any pair of types. Deciding subtyping for arbitrary types is equivalent to decide whether a type is equivalent to (that is, it has the same interpretation as) $\mathbb{0}$:

$$s \leq_{\mathcal{B}} t \Leftrightarrow [\![s]\!]_{\mathcal{B}} \subseteq [\![t]\!]_{\mathcal{B}} \Leftrightarrow [\![s]\!]_{\mathcal{B}} \cap \overline{[\![t]\!]_{\mathcal{B}}} = \varnothing \Leftrightarrow [\![s \wedge \neg t]\!]_{\mathcal{B}} = \varnothing \Leftrightarrow s \wedge \neg t = \mathbb{0}.$$

By using the definition of $\mathcal{B}[\![\ ]\!]$, we can show that every type is equivalent to a finite union where each summand is either of the form:

$$\Big(\bigwedge_{s \times t \in P} s \times t\Big) \wedge \Big(\bigwedge_{s \times t \in N} \neg(s \times t)\Big) \tag{4}$$

or of the form

$$\Big(\bigwedge_{s \to t \in P} s \to t\Big) \wedge \Big(\bigwedge_{s \to t \in N} \neg(s \to t)\Big) \tag{5}$$

Put $s \wedge \neg t$ in this form. Since it is a finite union, then it is equivalent to $\mathbb{0}$ if and only if each summand is so. So the decision of $s \leq_{\mathcal{B}} t$ is reduced to the problem of deciding whether the types in (4) and (5) are empty. The subtyping algorithm, then, has to work coinductively, decomposing these problems into simpler subproblems where the topmost type constructors have disappeared. In particular, in [Fri04] it is proved that the type in (4) is equivalent to $\mathbb{0}$ if and only if for every $N' \subseteq N$:

$$\left(\bigwedge_{(t \times s) \in P} t \ \wedge \bigwedge_{(t' \times s') \in N'} \neg t'\right) \simeq \mathbb{0} \ \text{ or } \left(\bigwedge_{(t \times s) \in P} s \ \wedge \bigwedge_{(t' \times s') \in N \setminus N'} \neg s'\right) \simeq \mathbb{0}; \tag{6}$$

while the type in (5) is equal to zero if and only if there exists some $(t' \to s') \in N$ such that for every $P' \subseteq P$:

$$\left(t' \wedge \bigwedge_{(t \to s) \in P'} \neg t\right) \simeq \mathbb{0} \ \text{ or } \left(\bigwedge_{(t \to s) \in P \setminus P'} s \wedge \neg s'\right) \simeq \mathbb{0}. \tag{7}$$

[3] For the details of the definition of the interpretation in the presence of recursive types, the reader is invited to consult [Fri04] and [FCB02]. The construction is also outlined in [CF05].