Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 322 (2016) 135–151

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Partial and Complete Processes in Multiparty Sessions [1]

Mario Coppo Mariangiola Dezani-Ciancaglini
Ines Margaria Maddalena Zacchi [2]

Dipartimento di Informatica Università di Torino, corso Svizzera 185, 10149 Torino, Italy

Abstract

Multiparty sessions describe the interactions among multiple agents in a distributed environment and require essentially two steps: the specification of the communication protocols and the implementation of such protocols as processes. Multiparty session types address this methodology: global and session types provide the communication protocols, whereas the processes describe the behaviour of the peers involved in the sessions. Because of the close relationships between types and processes, some information, such as the names of senders and receivers, are replicated both in types and in processes.
In multiparty conversations it is quite natural that participants with essentially the same role are implemented by processes that follow the same pattern, differing only in the senders and receivers of communication actions. In order to allow for a lighter and less rigid development of processes, we propose a translation tool which allows one to write processes in a simplified syntax, called partial syntax, where the names of senders/receivers for input/output operations are omitted. By adding the missing information, *partial processes* can be automatically translated in *complete processes*, for which an operational semantics is defined. The partial syntax, in particular, allows one to use the same process template to implement similar participants.
In this paper we present a translation and type checking algorithm from partial to complete processes, which, if successful, also assures that the target processes are well typed. The algorithm is synthesised from a rule-based description of the translation in natural semantics and it is proved sound and complete with respect to the translation rules.

Keywords: π-calculus, session types, multiparty sessions.

1 Introduction

Session types are one of the most successful formalisms introduced to describe communicating processes and to study their behaviour. The basic idea, appeared first in [11] and [6], is to introduce a new form of polymorphism which permits the typing of channel names by structured sequences of types, abstractly representing the trace

[1] This work was partially supported by ICT COST Action IC1201 BETTY, MIUR PRIN Project CINA Prot. 2010LHT4KM and Torino University/Compagnia San Paolo Project SALT.

[2] Email: {coppo,dezani,ines,zacchi}@di.unito.it

http://dx.doi.org/10.1016/j.entcs.2016.03.010

of the usage of the channels. In modelling distributed systems, where processes interact by means of message passing, it is appropriate to allow many interactions to occur within the scope of private channels following disciplined protocols. As usual, we call sessions such private interactions and session types the protocols describing them. In its simplest form, a session is established between two peers, such as a client connecting with a server. In general, a session may involve any (usually fixed) number of peers. In these cases, we speak of multiparty sessions and of multiparty session types [7] for their protocol descriptions. A multiparty session type theory consists of three parts: global types, processes, and local types, called also session types. *Global types* describe communication protocols in terms of the interactions between peers, of the order of these interactions, and of the kind of exchanged messages. The description given by a global type is neutral, independent from the peers and their viewpoints. *Processes* describe, by means of a formal language, the behaviour of the peers involved in the session. For each peer a *session type* describes the same communication protocol as the global type, but from the viewpoint of the peer. Local and global types are related by a projection operation that extracts local types from the global ones, and a type system makes sure that a process uses the communication channels it owns according to their local types. Among the more interesting features of interactions between peers, session delegation is a key operation, which permits to rely on other parties for completing specific tasks transparently, in a type safe manner. A typical scenario is given by the protocol of Remote Procedure Call for server/client distributed systems. In such a case the server, after receiving a request from a client, delegates remaining interactions with the client to an application process. The client and the application process are initially unknown to each other, but later communicate directly, transparently to the client, through dynamic mobility of the communication channel. Such protocol is usually synchronous, because the processes involved in the communications remain tightly coupled.

Regarding the syntax of processes, the need to specify sender and receivers for each input/output operation makes cumbersome the code writing; moreover, in distributed applications often many processes perform exactly the same pattern of input/output operations, differing only for the involved participants. For example in the Remote Procedure Call protocol, the application processes, that provide the services, in many interesting cases differ only in the process names involved in the communications. In this paper we present a translation algorithm that allows one to code processes in a simplified syntax, called *partial syntax*, which does not require to specify the names of senders/receivers for input/output operations. Partial code is simpler to write and can be shared by different participants in a conversation, but it is incomplete and cannot be directly executed. The executable code of the processes, written in *complete syntax* (i.e. including the names of the processes involved in the communications) can be obtained automatically from the partial code by exploiting the information given by the global types. The translation is successful only if the target process is well typed according to standard typing rules for multiparty sessions, so the translation algorithm also includes type checking.