The Intimacy of Session Types and Proof Theory
Recent developments

Bogdan Aman[1], Gabriel Ciobanu[1] and Ross Horne[1,2]

[1] Romanian Academy, Institute of Computer Science, Iaşi, Romania
[2] School of Computer Engineering, Nanyang Technological University, Singapore
bogdan.aman@gmail.com, gabriel@info.uaic.ro, rhorne@ntu.edu.sg

Session types were originally inspired by linear logic. The computational interpretation of (intuitionistic) linear logic due to Abramsky [1] decomposes choice into internal and external choice, which interpret a de Morgan dual pair of additive operators. Honda exploited this duality when devising a theory of binary session types [7]. However, the correspondence between linear logic and session types is not exact. In particular, Honda's binary session types feature a self-dual non-commutative operator representing sequentiality that is not present in linear logic. More recently, proof theorists devised a framework, called the calculus of structures [6], that is capable of expressing extensions of linear logic with a self-dual non-commutative operator. This advance enabled the authors to devise a tight and direct formulation of finite multi-party session types [9] as propositions in a proof calculus called MAV [2, 10].

A significant fragment of the session modelling language Scribble [8] can be embedded in the proof calculus MAV. The behaviour of each party in a session is modelled by local types, which we ensure are *compatible* [3] in the sense that all parties realise a global protocol when executed together. Remarkably, executions witnessing that local types are compatible are exactly proofs in the system MAV. Thus *provability* in MAV corresponds with *multi-party compatibility*. Since MAV is an analytic proof calculus this correspondence provides procedures for deciding multi-party compatibility.

Further to capturing multi-party compatibility, proofs in MAV can be used to define a subtype system. In typical applications of session types for static analysis [12, 13], the control flow of a program is extracted and the values communicated are replaced by types to obtain a session type. We then check that the session type obtained from the control flow is a subtype of the specification. As in the binary session case [14, 4], subtyping can vary branches of a choice and the basic types input and output on channels. Subtyping also permits causal dependencies to be relaxed. The subtyping relation is given by exactly the provable linear implications in MAV. A corollary of the cut elimination result for MAV [10] is that linear implication defines a preorder over session types.

Reassuringly, as a process preorder, linear implication is sound with respect to both (weak complete) simulation and pomset traces [5]; hence respects branching time and causality. A subtype system based on linear implication is therefore safe to use for processes deployed on systems guaranteeing a consistency model at least as strong as causal consistency. Hence when linear implication is employed for subtyping, false positives where subtyping incorrectly claims a party meets

its specification are less likely than for definitions of subtyping based on trace inclusion for instance.

Correspondences between linear logic and session types have been sought as an objective justification for design decisions in session types. We have demonstrated an intimate correspondence where propositions in MAV are session types, proofs are witnesses of multi-party compatibility, and linear implication is subtyping. By extending with first-order quantifiers to obtain MAV1 [11], processes can be modelled in a similar fashion by directly embedding processes as predicates. A novel feature of MAV1 is a pair of de Morgan dual nominal quantifiers that model private names in the π-calculus. MAV1 can embed numerous extensions of finite π-calculus processes such as a π-calculus with both internal and external choice in which multi-party compatibility can be extended to the process level. Future work includes establishing the consistency of second-order extensions of MAV for modelling fixed points and delegation.

References

1. Samson Abramsky. Computational interpretations of linear logic. *Theoretical computer science*, 111(1):3–57, 1993.
2. Gabriel Ciobanu and Ross Horne. Behavioural analysis of sessions using the calculus of structures. In *PSI 2015, 25-27 August, Kazan, Russia*, volume 9609 of *LNCS*, 2015.
3. Pierre-Malo Deniélou and Nobuko Yoshida. Multiparty compatibility in communicating automata: Characterisation and synthesis of global session types. In *Automata, Languages, and Programming*, pages 174–186. Springer, 2013.
4. Simon Gay and Malcolm Hole. Subtyping for session types in the pi calculus. *Acta Informatica*, 42(2-3):191–225, 2005.
5. Jay L. Gischer. The equational theory of pomsets. *Theoretical Computer Science*, 61(2-3):199–224, 1988.
6. Alessio Guglielmi. A system of interaction and structure. *ACM Transactions on Computational Logic*, 8(1), 2007.
7. Kohei Honda. Types for dyadic interaction. In *CONCUR'93*, pages 509–523. Springer, 1993.
8. Kohei Honda et al. Scribbling interactions with a formal foundation. In *ICDCIT 2011*, volume 6536 of *LNCS*, pages 55–75. Springer, 2011.
9. Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. *Journal of the ACM (JACM)*, 63(1):9, 2016.
10. Ross Horne. The consistency and complexity of multiplicative additive system virtual. *Sci. Ann. Comp. Sci.*, 25(2):245–316, 2015.
11. Ross Horne, Alwen Tiu, Bogdan Aman, and Gabriel Ciobanu. Private names in non-commutative logic. In Josée Desharnais and Radha Jagadeesan, editors, *CONCUR 2016*, number 31 in LIPIcs, pages 1–16.
12. Raymond Hu, Nobuko Yoshida, and Kohei Honda. Session-based distributed programming in java. In *ECOOP*, pages 516–541. Springer, 2008.
13. Nicholas Ng, Nobuko Yoshida, and Kohei Honda. Multiparty session C: Safe parallel programming with message optimisation. In *Objects, Models, Components, Patterns*, pages 202–218. Springer, 2012.
14. Benjamin Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. In *LICS'93*, pages 376–385. IEEE, 1993.