ABCD in Edinburgh

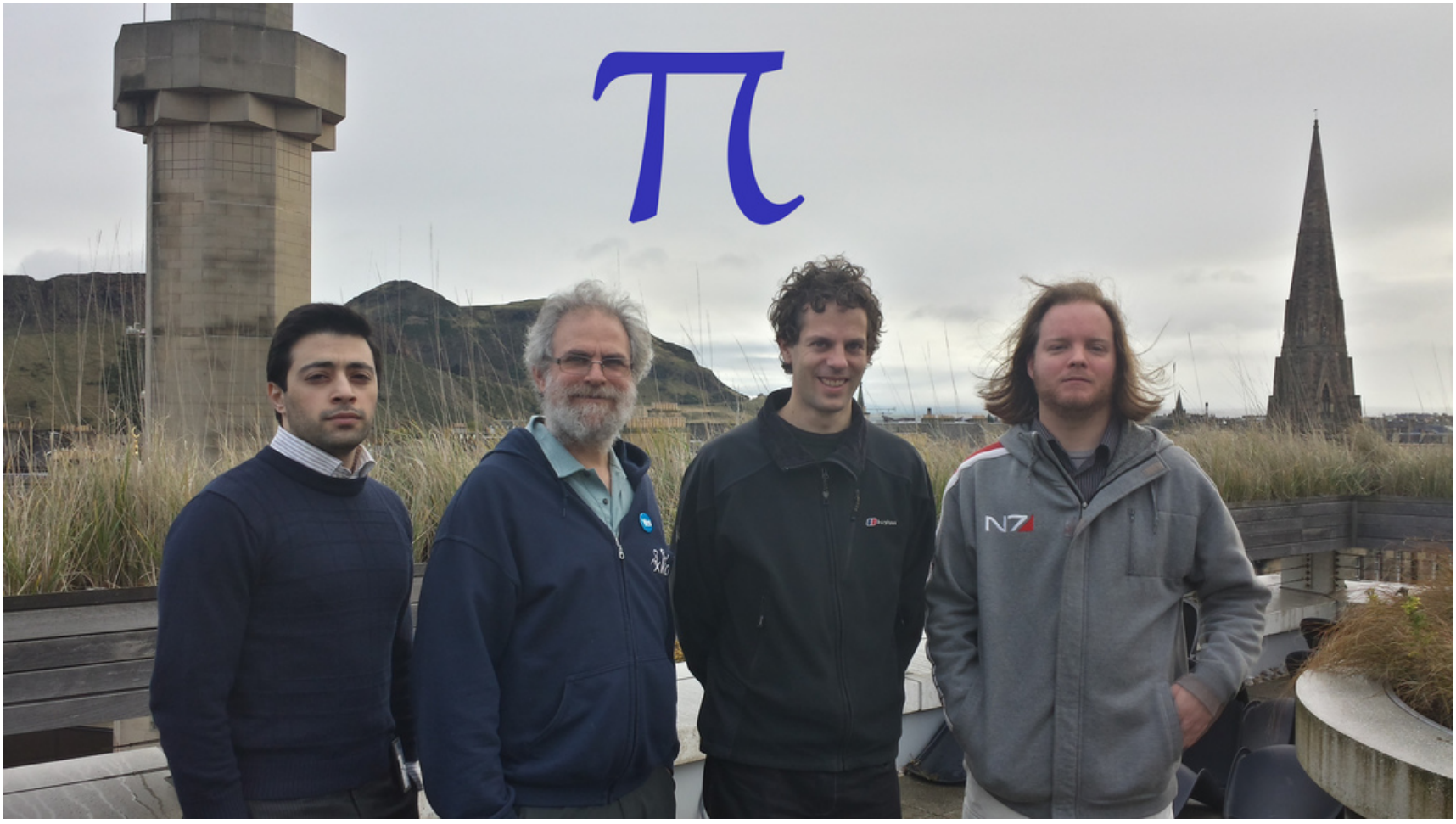
Sam Lindley

Laboratory for foundations of computer science
The University of Edinburgh

Sam.Lindley@ed.ac.uk

9th January 2014

People



Shayan Najd, Philip Wadler, Sam Lindley, Garrett Morris

Work packages

- P1: Industrial case studies
  WP1.1: Amazon Web Services. WP1.3: OOI cyberinfrastructure. WP1.5: Review & organisation.
- P4: Mainstream programming languages
  WP4.2: Gradual session types in Python.
- **P5: Web applications**
  WP5.1: Session types in Links. WP5.2: Session types for distribution. WP5.3: Reliability and recovery. WP5.5: Case studies and empirical evaluation. WP5.6: Infrastructure. WP5.7: Tutorial and dissemination.
- P6: Environments, modelling & empirical studies
  WP6.1: Environments. WP6.2: Modelling. WP6.3: Empirical studies.
- **P7: Foundations of session types**
  WP7.1: Races and deadlock. WP7.2: Multiparty session types. WP7.3: Productive streams.

Propositions as sessions (Wadler, 2012)

Caires and Pfenning (2010) gave a Curry-Howard correspondence for session types through πDILL, a process calculus for intuitionistic linear logic.

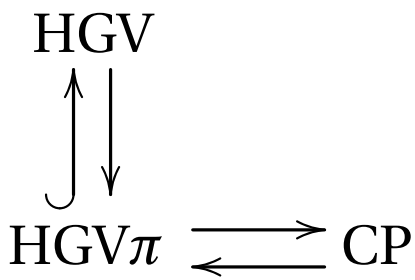
propositions	$\Longleftrightarrow$	session types
proofs	$\Longleftrightarrow$	processes
cut elimination	$\Longleftrightarrow$	communication

Wadler adapted this idea to classical linear logic.

- CP: a process calculus for classical linear logic
- GV: a functional programming language with session types (based on Gay and Vasconcelos, 2010)
- GV $\longrightarrow$ CP
- No translation from CP to GV

Sessions as propositions (Lindley and Morris, 2014)

- HGV = GV + missing features
- HGVπ is the session-typed fragment of HGV
- Translations



- CP, HGV, and HGVπ are all equally expressive

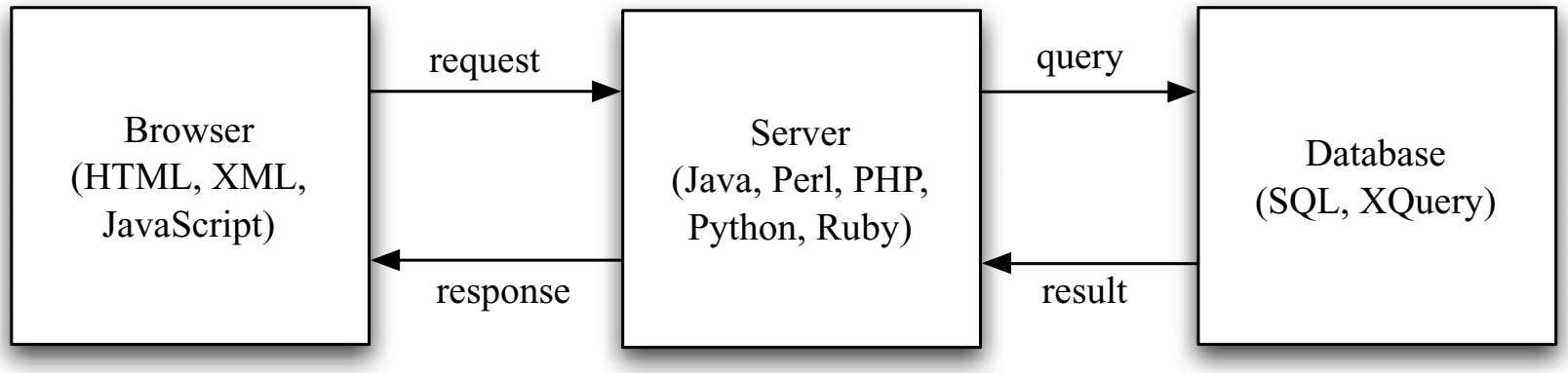
We have prototype implementations of CP and HGV + various extensions. Implementations use big-step semantics.

Concurrency

- CP is deterministic: it fails to capture many interesting forms of concurrency
- Goal: add features to CP that allow us to embed π-calculus while maintaining the connection to classical linear logic
- One natural way to add non-deterministic concurrency to HGV is via *access points* (Gay and Vasconcelos, 2010)
- Not clear if these correspond to anything meaningful in CP
- Garrett will talk more about this later

Links

Typical web applications



Links

- Statically typed functional programming language for the web
- Single source language targets browser, server, and database
- Features include: strict, type inference, row types, effect types, formlets, language integrated query, first-class continuations, statically typed asynchronous message passing concurrency

Adding session types to Links

Challenges

- Combining type inference with linear types
- Supporting the existing concurrency model on top of session types

Promising approaches

- Stratify the language
  - Only treat linearity in the session-typed fragment of the language
  - Inspiration: adjoint calculus (Benton and Wadler, 1996), FRP (Krishnaswami and Benton, 2011), linear contextual monad (Toninho, Caires, and Pfenning, 2013)
- Local type inference
  - avoid problems of full type inference for linear types

What next?

- Embedding of π-calculus in as small an extension of CP as we can get away with
- Converge on initial design for session types in Links
- Initial implemention of session types for Links
- NII Shonan meeting on software contracts for communication, monitoring, and security (May 2014)
- Tutorial on session types in Links at the Advances in Programming Languages summer school (August 2014)

Other stuff

- Blame (Wadler)
- Feldspar (Najd)
- Handlers for algebraic effects (Lindley)
- Language integrated query (Lindley and Wadler)
- Instance chains (Morris)