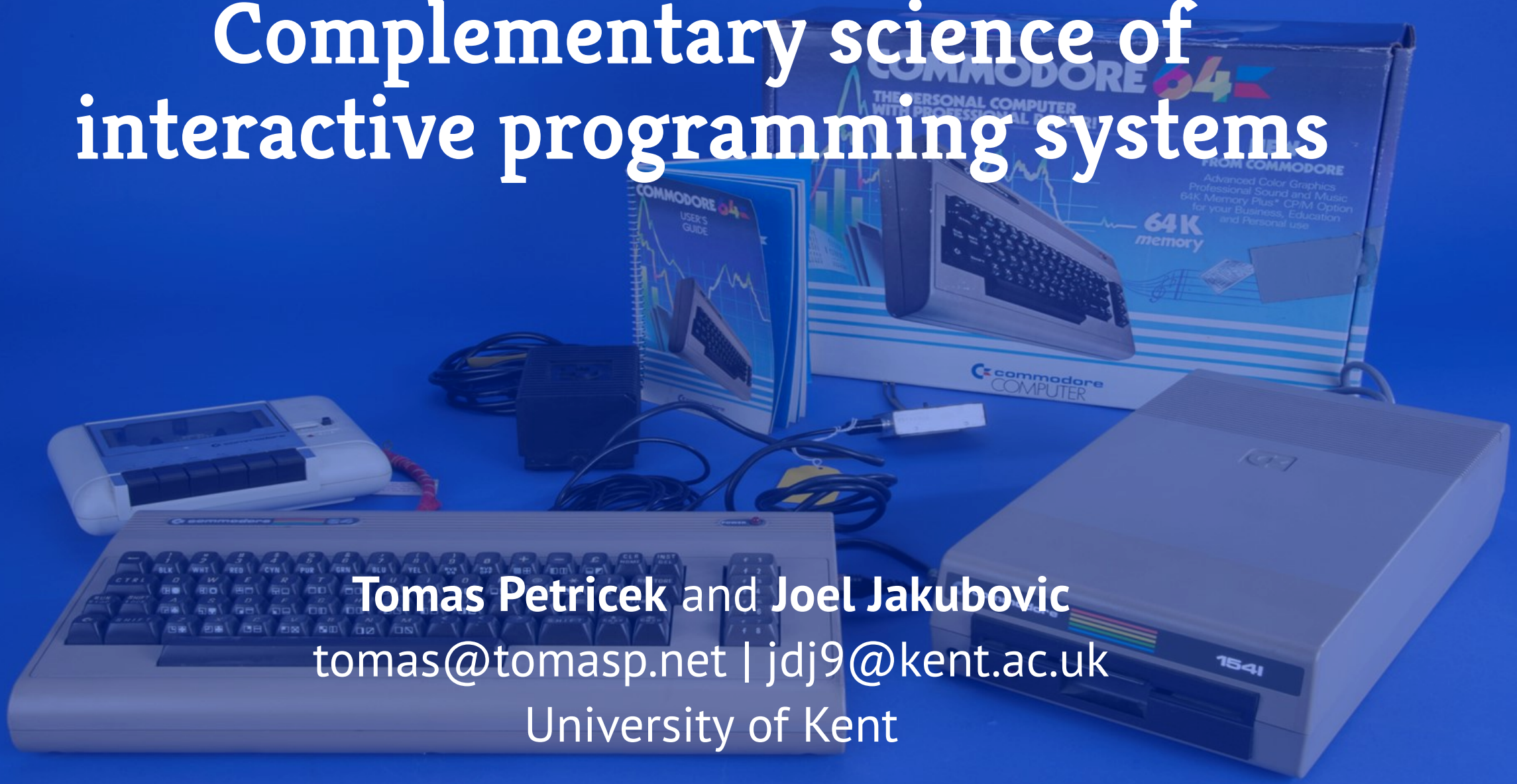
Complementary science of interactive programming systems



Tomas Petricek and **Joel Jakubovic**
tomas@tomasp.net | jdj9@kent.ac.uk
University of Kent

Paradigm shift in computer programming

Programming languages

3.1. VARIABLES

3.1.1. Syntax

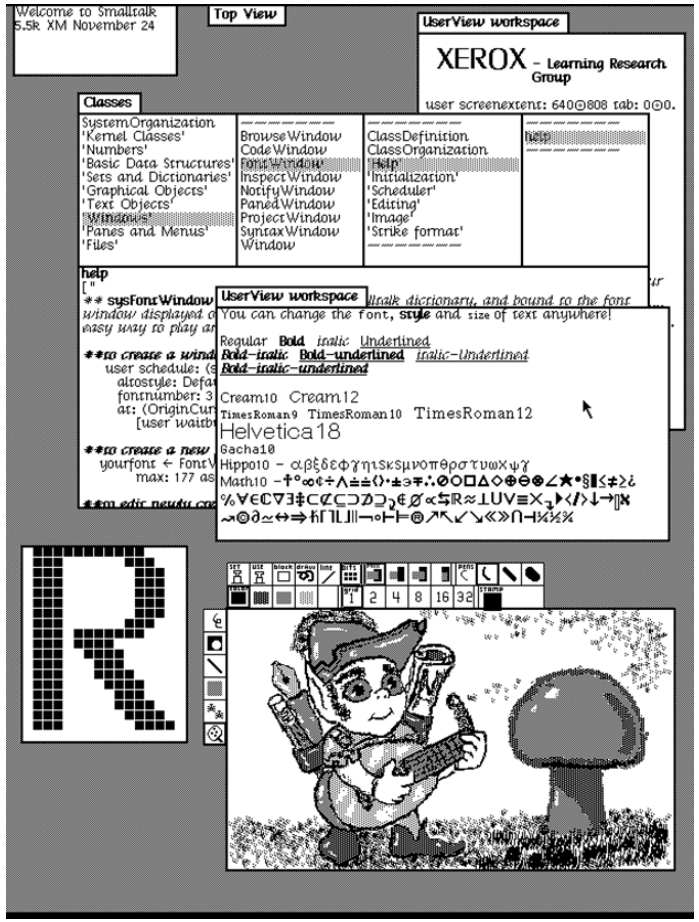
⟨variable identifier⟩ ::= ⟨identifier⟩
⟨simple variable⟩ ::= ⟨variable identifier⟩
⟨subscript expression⟩ ::= ⟨arithmetic expression⟩
⟨subscript list⟩ ::= ⟨subscript expression⟩|⟨subscript list⟩,
⟨subscript expression⟩
⟨array identifier⟩ ::= ⟨identifier⟩
⟨subscripted variable⟩ ::= ⟨array identifier⟩[⟨subscript list⟩]
⟨variable⟩ ::= ⟨simple variable⟩|⟨subscripted variable⟩

3.2. FUNCTION DESIGNATORS

3.2.1. Syntax

⟨procedure identifier⟩ ::= ⟨identifier⟩
⟨actual parameter⟩ ::= ⟨string⟩|⟨expression⟩|⟨array identifier⟩|
⟨switch identifier⟩|⟨procedure identifier⟩
⟨letter string⟩ ::= ⟨letter⟩|⟨letter string⟩⟨letter⟩
⟨parameter delimiter⟩ ::= ,|)⟨letter string⟩ :(
⟨actual parameter list⟩ ::= ⟨actual parameter⟩|
⟨actual parameter list⟩⟨parameter delimiter⟩
⟨actual parameter⟩
⟨actual parameter part⟩ ::= ⟨empty⟩|(⟨actual parameter list⟩)
⟨function designator⟩ ::= ⟨procedure identifier⟩
⟨actual parameter part⟩

Programming systems



Paradigm shift in computer programming

Programming languages

- Algol, Java, Haskell
- Code in a formal language
- Program as text
- Can be analyzed and run

Programming systems

- Interlisp, Smalltalk, Hypercard
- State includes code and data
- Interesting user interfaces
- Can be interacted with

Kuhn loss

What do we lose if we think about programming languages rather than programming systems?

What is using past programming system like?

Harder to study than programming languages!

Do you need a real machine to experience the interaction?

What can we learn from past programming systems?

Complementary science of programming

Hasok Chang (2012)

- Look at history to recover forgotten knowledge
- Contribute alternative views to current science
- Physics ideas abandoned due to experimental failures

Programming systems

- Many innovative past systems worth exploring!
- Abandoned not just for scientific reasons
- Easier to recreate and further develop than in physics

Commodore 64 BASIC

Recreating the interactive experience of programming Commodore 64

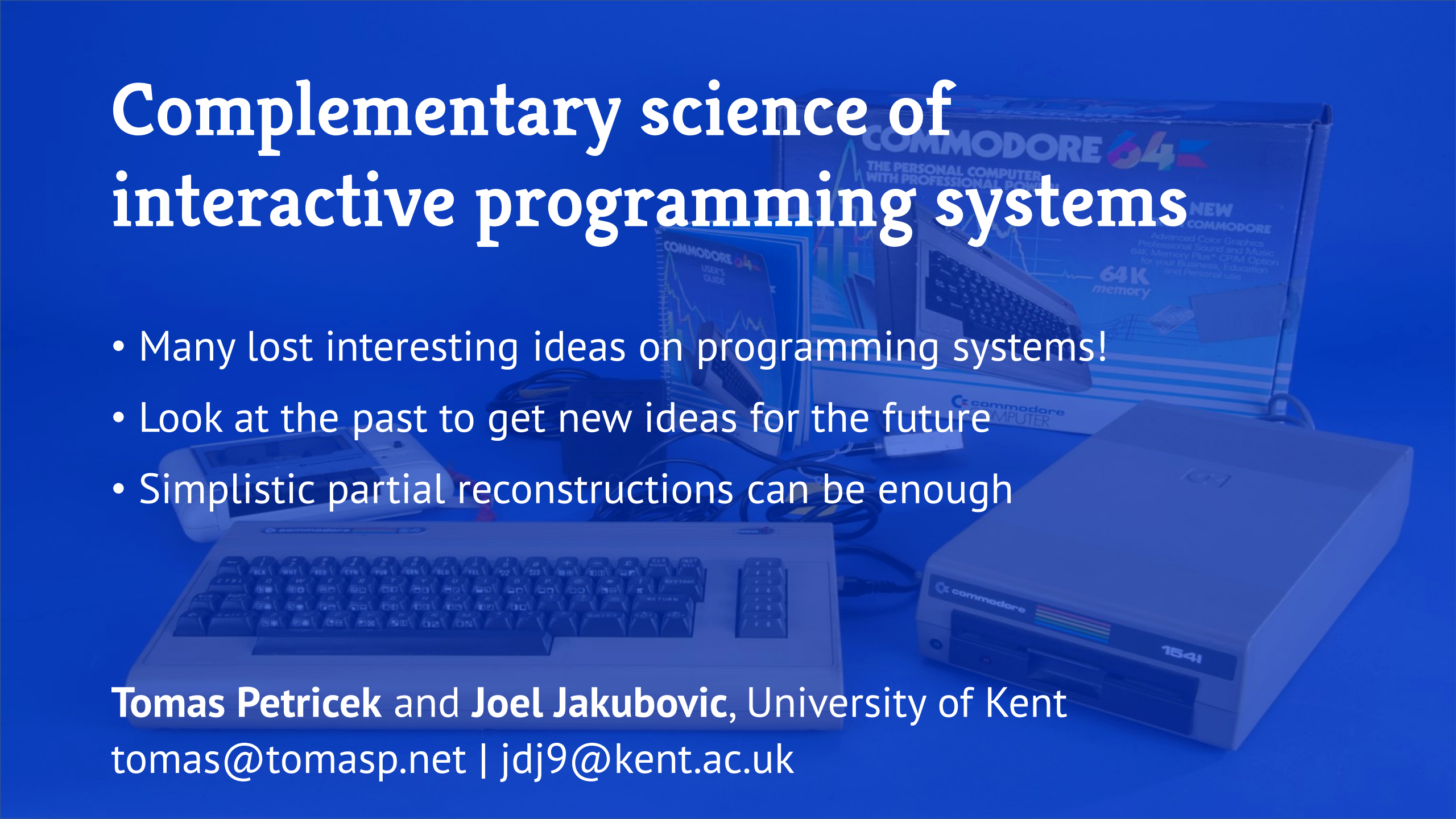
What makes C64 BASIC interesting?

Everything done through one kind of interaction

Encourages users to become programmers

Memory mapped I/O with POKE offers hacker flexibility

Complementary science of interactive programming systems



- Many lost interesting ideas on programming systems!
- Look at the past to get new ideas for the future
- Simplistic partial reconstructions can be enough

Tomas Petricek and **Joel Jakubovic**, University of Kent
tomas@tomasp.net | jdj9@kent.ac.uk