A Proposal to use ABS as a Proof of Concept Platform for New Theories

S. Lizeth Tapia Tarifa

University of Oslo, Norway

sltarifa@ifi.uio.no

TU Darmstadt - Germany, May 2018

ABS and its Application's Domain

- Distributed software application
  (e.g., cloud computing, distributed work processing)
- Cyber physical systems
  (e.g., software updates for cars)
- Distributed business/operational processes
  (e.g., operational planning, railway operations)
- Formal systems
  (e.g, OS: multicore data access, memory models)
- Biological systems
- ...

The Flexible framework of ABS

So far ABS has been able to closely represent the intended domain

- User defined data types and functions:
  allows to express and manipulate data for various domains
- Synchronous and asynchronous communication:
  helps to naturally describe interactions between objects/components
- Cooperative scheduling:
  naturally describe concurrent workflows.

ABS as a Proof of Concept Platform

Can we use ABS as proof of concept platform for new theories?

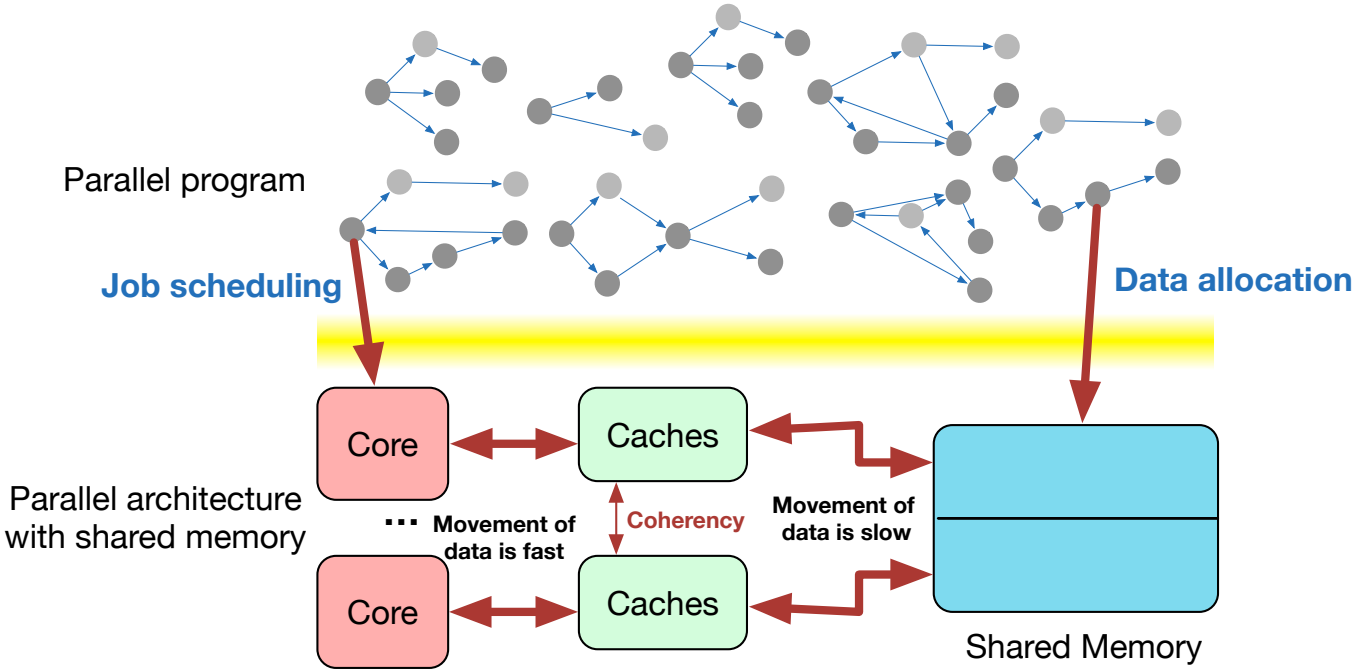
Proof of Concept (POC)

- Demonstrate how a new concept/theory has the potential to be applied to real applications
- Test new concept/theory under certain assumptions and demonstrate their functionality
- Observe the functionality of a concept/theory when it is integrated into a model of an existing system
- Explore an emerging concept/theory and provide evidence to the potential stakeholders

Built an ABS executable model that gives an idea how a theory/concept could potentially work

A Concrete New Concept: Location Types

Problem Domain:



- Unnecessary movement of data affects performance
- Disconnection between locations for processing and locations of data

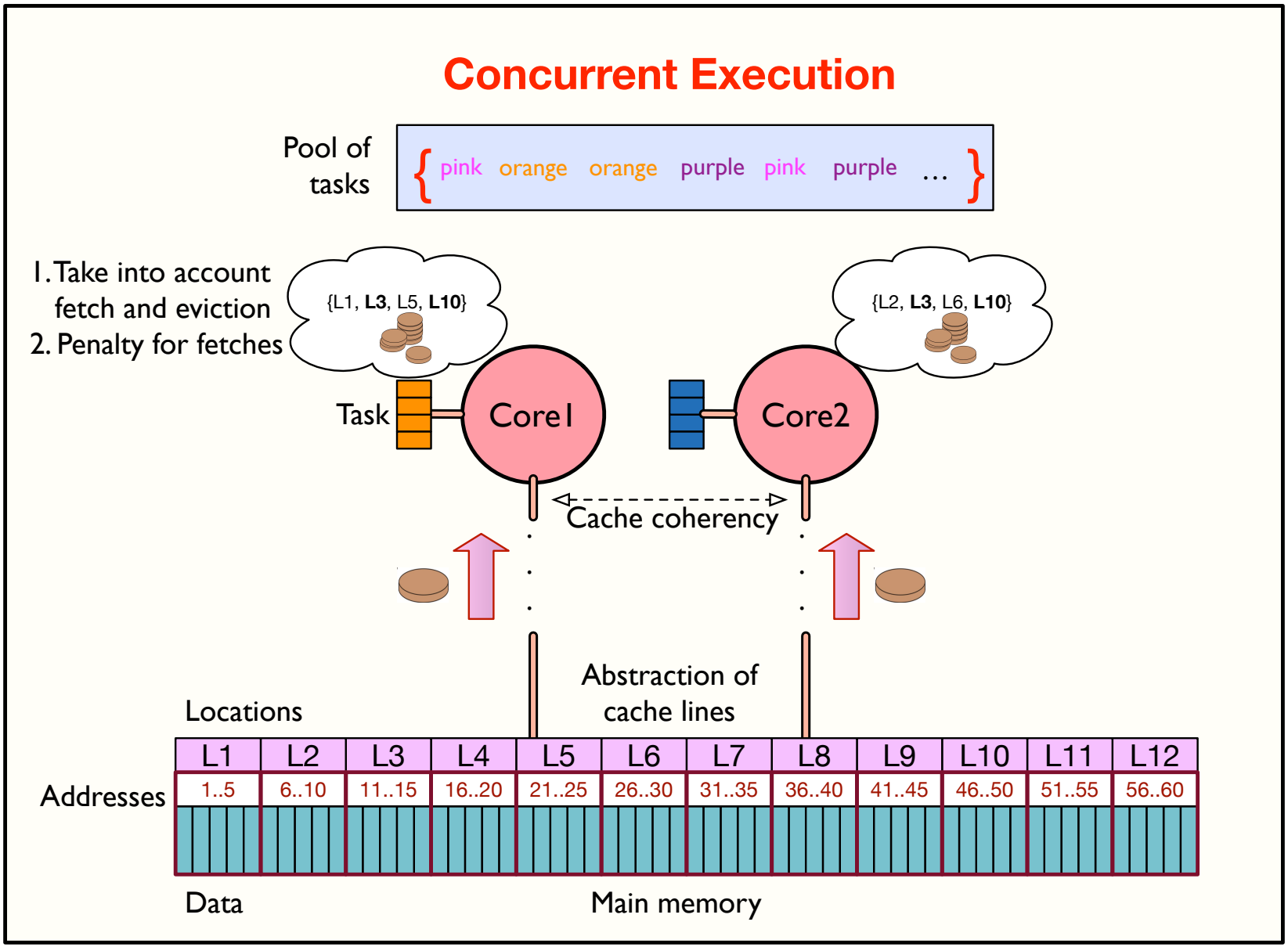
Abstract representation of the main memory

Location		L_x			L_y			L_z			
Address	...	0x11cedf	...	0x11cee2	0x11cee3	...	0x11cee5	0x11cee6	...	0x11cee8	...
Value											

Memory footprints:

- Use the idea of abstract locations in main memory to approximate reads and writes data access
- Develop a type system that uses locations to statically extract and describe how workflows interact with Memory

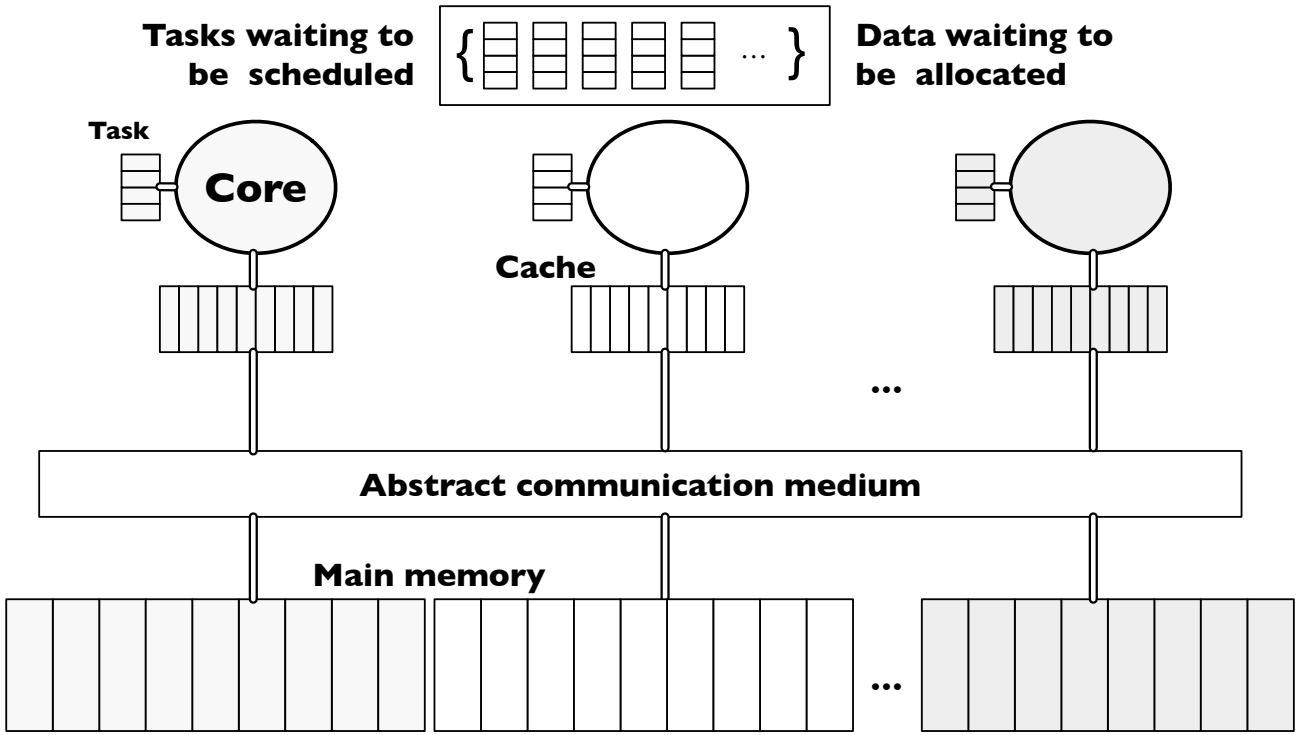
Type system to predict data accesses



Type Analysis

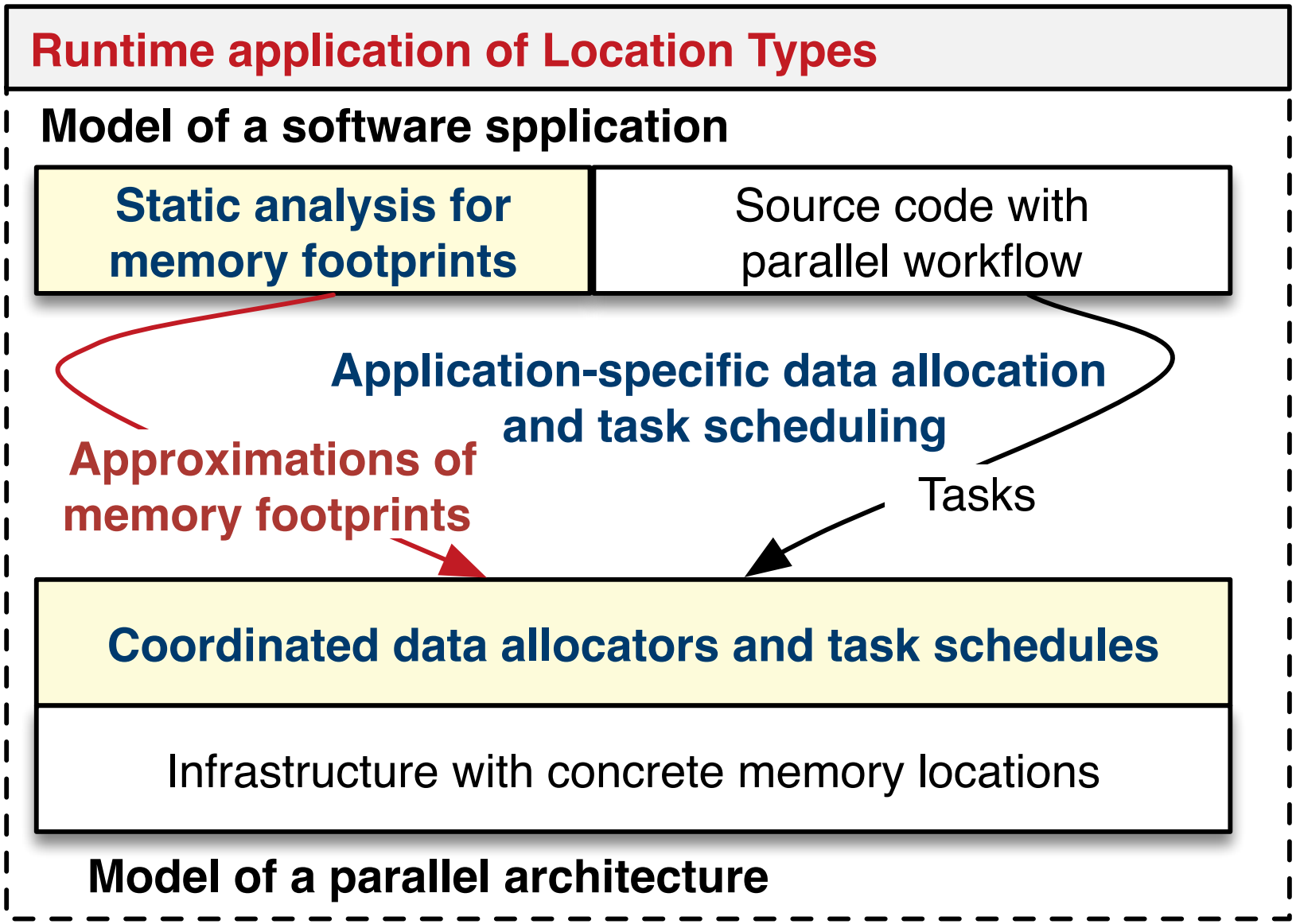
- Standard type system
  - Variables, pointers and addresses are in the right locations
  - Check that all locations are understood in the different tasks/processes
  - Check that references of variables are contained in only one location in main memory
  - Check that the state of the local data complies with the state in the main memory
- Advance behavioural types – "memory access footprints"
  - Check that the runtime system accesses locations as expected during execution (e.g., reading/writing)
  - Check that the cache memory (set of locations), changes as expected during execution
- Types at runtime - model based scheduling and allocation decisions
  - Can we make use of memory access footprints for scheduling and allocation?

Starting point in ABS: A multicore layer of execution with coherent caches and shared memory



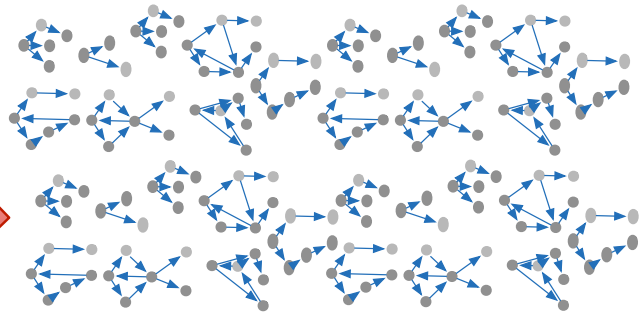
Can we build and use this layer as an API similar to the cloud API?

Approach

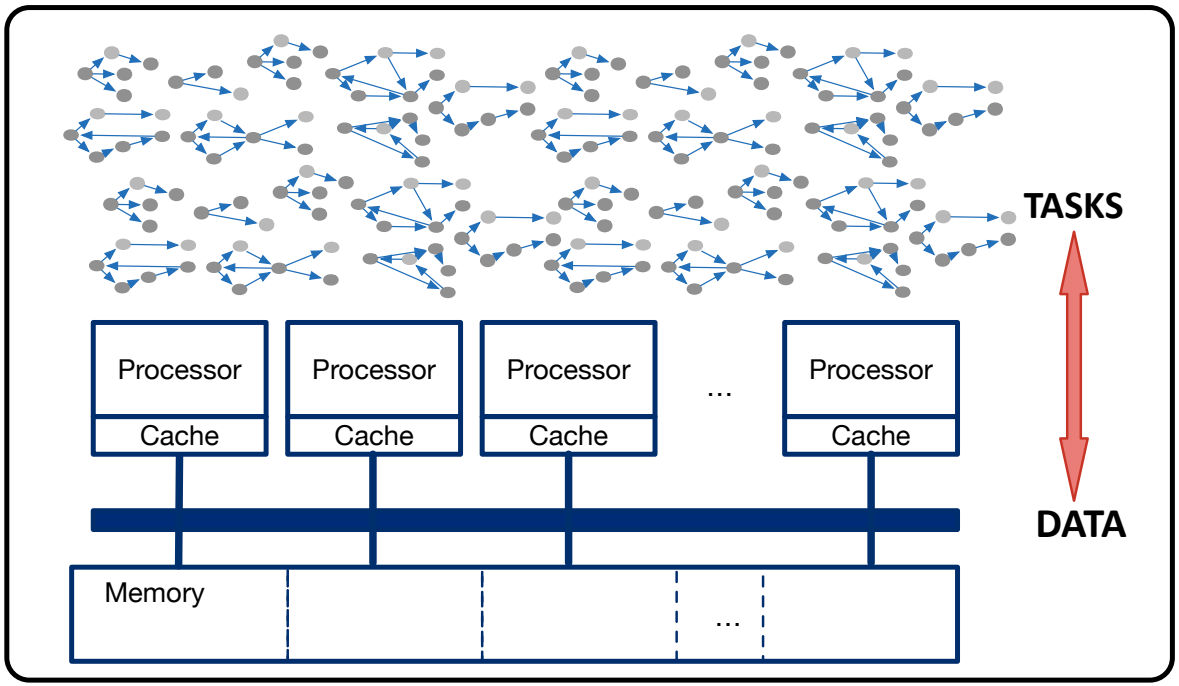

Runtime application of Location Types
Model of a software spplication
Static analysis for memory footprints
Source code with parallel workflow
Application-specific data allocation and task scheduling
Approximations of memory footprints
Tasks
Coordinated data allocators and task schedules
Infrastructure with concrete memory locations
Model of a parallel architecture

Validation: simulations with measurements

Abstract



Collect measurements during simulations for comparisons



ABS Extensions?

- What is missing: Visualisation targeting multicore execution
- What is challenging: how to relate the theory to the model?
  - Is it a one to one matching between the theory and the model?
  - can we develop a simulation/bisimulation relation method?
  - can we express it as properties to the proof system?

Summary

Can we use ABS as proof of concept platform
for location types and their use for schedulers and allocators?

Proof of Concept for *Location Types*

- Demonstrate how *Location Types* have the potential to be applied in parallel software
- Test *Location Types* under certain assumptions and demonstrate their functionality for schedulers and allocators
- Observe the functionality of *Location Types* when it is integrated into a model of a multicore architecture running parallel tasks.
- Explore *Location Types* and provide evidence to potential stakeholders

Summary

- What is missing:
  Visualisation in ABS targeting multicore execution
- What is challenging:
  how to relate the theory to the ABS model?
  - Is it a one to one matching between the theory and the model?
  - can we develop a simulation/bisimulation relation method?
  - can we express it as properties to the proof system?

- Are there other challenges?

THANK YOU