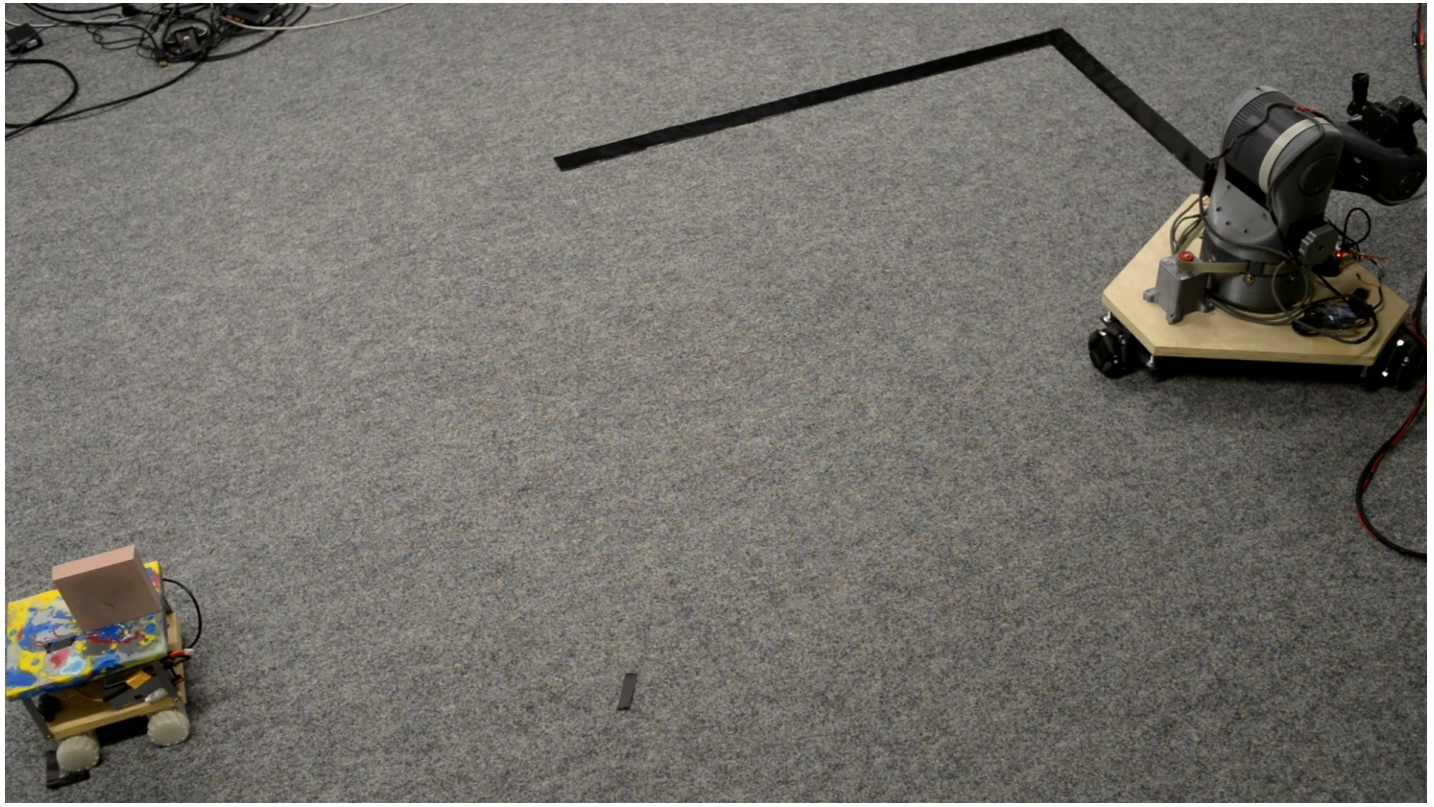
Motion Session Types for Robotic Interactions

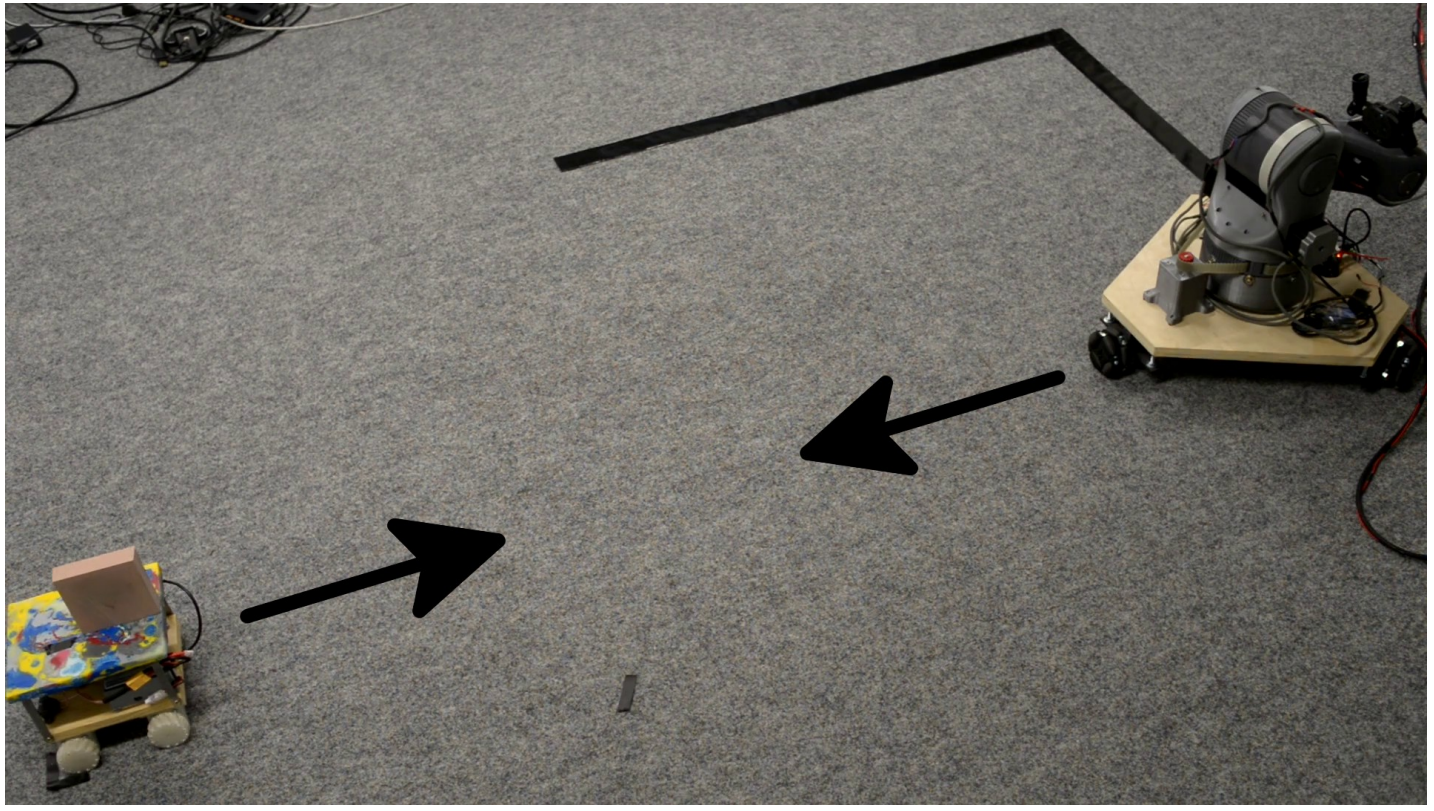
Rupak Majumdar	MPI-SWS
Marcus Pirron	MPI-SWS
Nobuko Yoshida	Imperial College London
Damien Zufferey	MPI-SWS

ECOOP 2019
18.07.2019

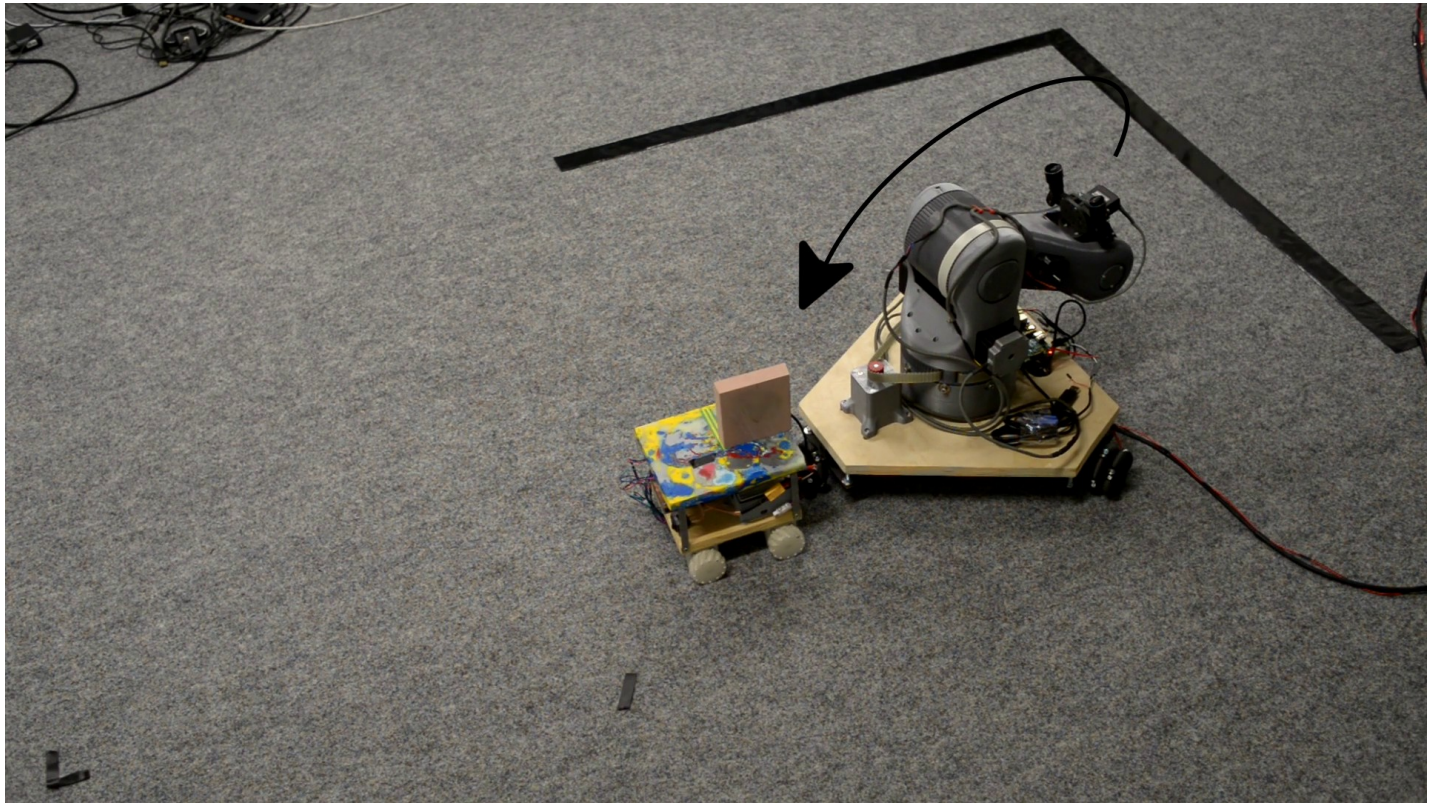
Example: Handover



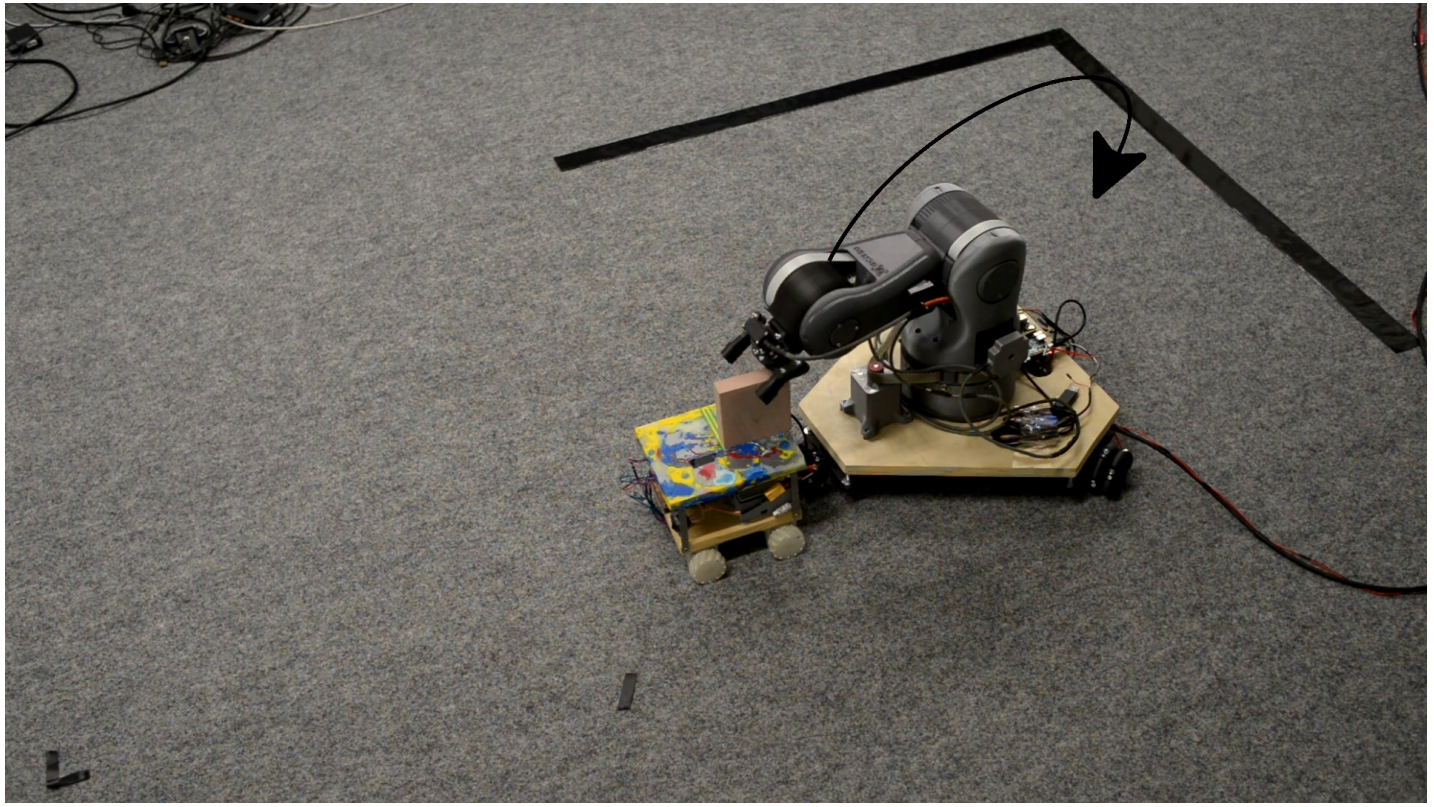
Example: Handover (Meet)



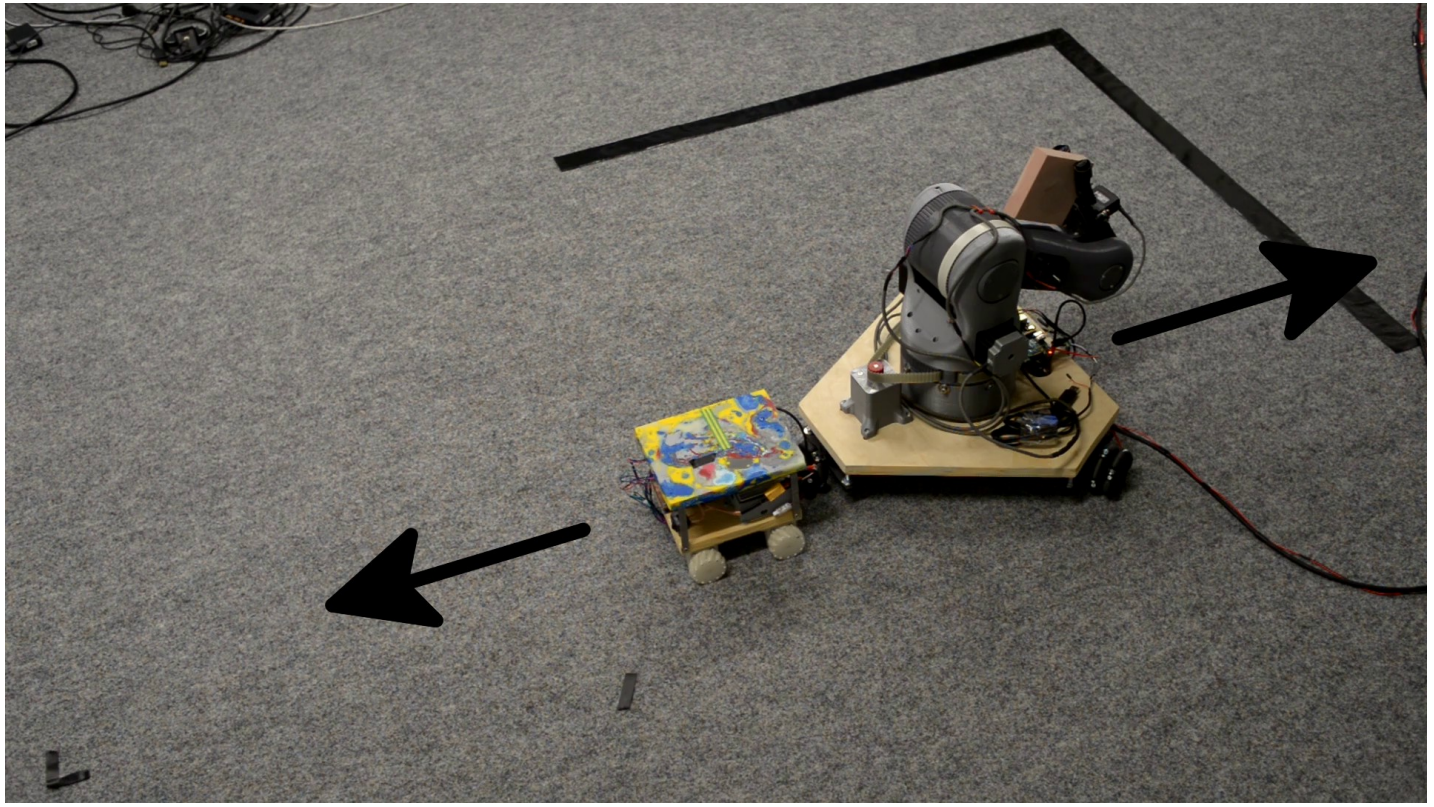
Example: Handover (Grab)



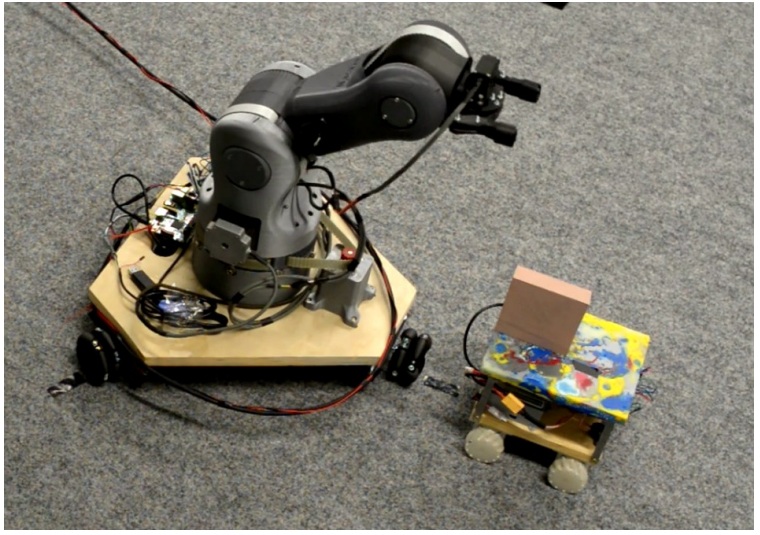
Example: Handover (Fold)



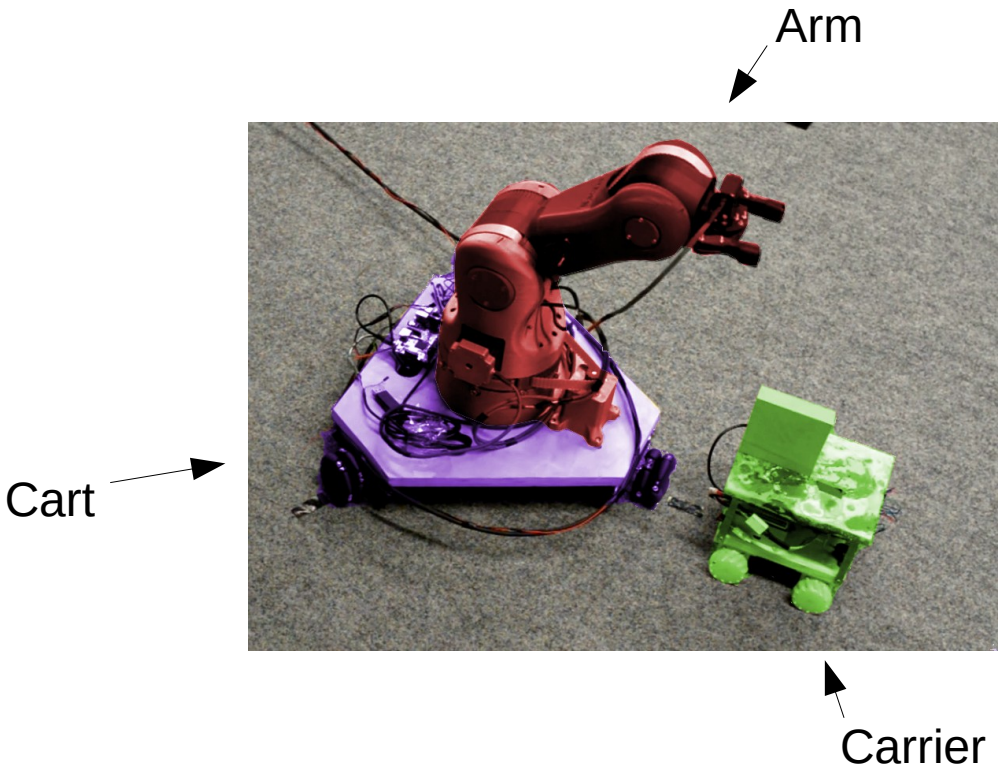
Example: Handover (Done)



How Many Robots?



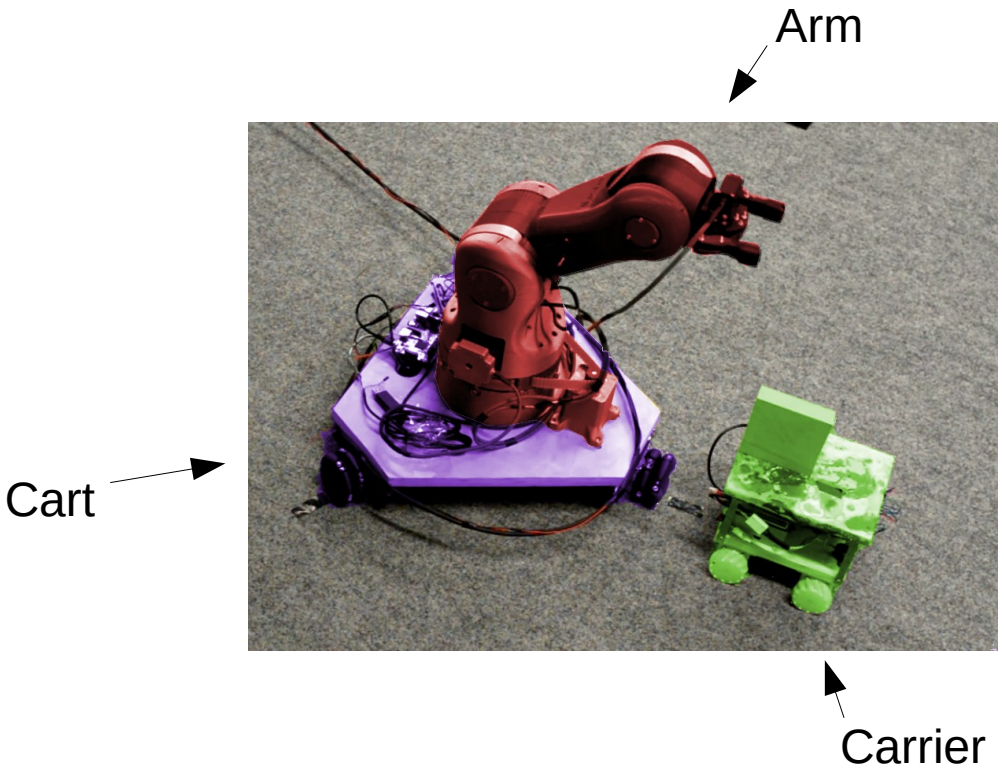
How Many Robots?

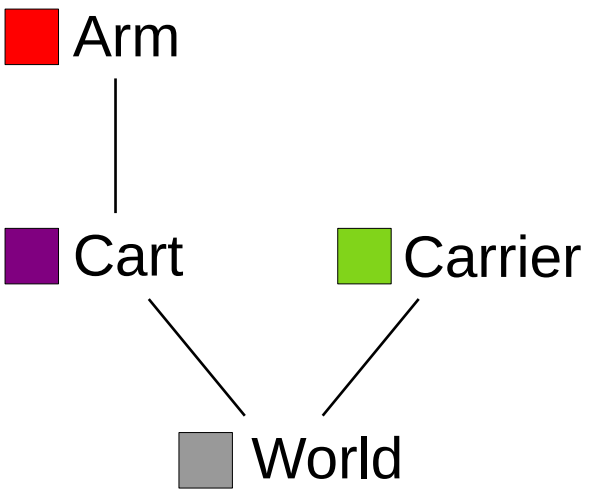


Cart & Arm:

Two robots attached together that act as “one” robot (communication)

How Many Robots?

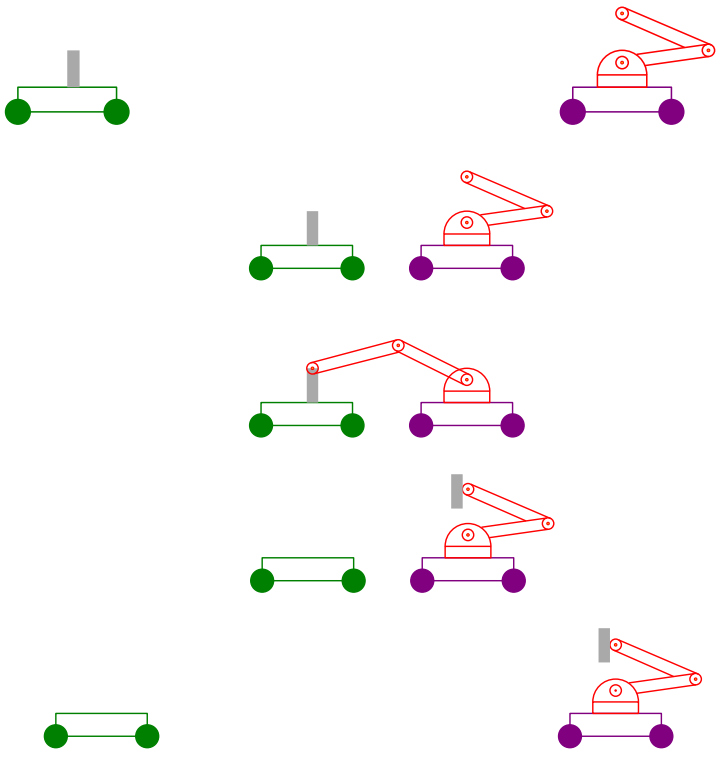




Robotic Program(mer)s as Target

- Message-Passing Abstraction
  - Robotic Operating System (ROS): publish-subscribe model
  - ROS is a de facto standard in academia (used for prototyping in industry)
- Modular robots
  - Software: logically separate units
  - Hardware: in a shared world (physical coupling)
- Different “kinds” of code
  - Planning: synchronize with other robots, decide what to do next, ...
  - Control: following trajectories, actuation, sensing, ...

Arm's Code



```
while true do
  receive (Cart, idle)
    grab(loc) ⇒
      grip(loc);
      send(cart, ok)
    fold ⇒
      move(origin);
      send(cart, ok)
    done ⇒
      break
```

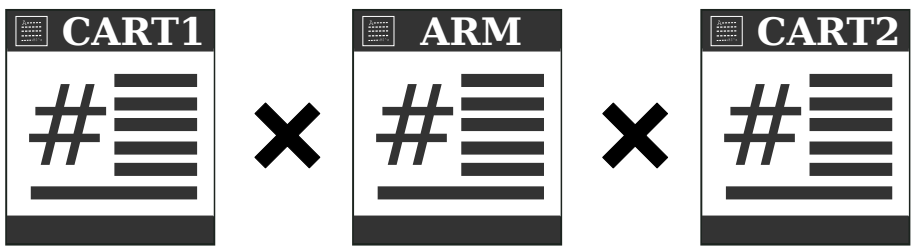
Motion primitive
Communication

Goal: Verification

Global Spec.

$\varphi(Cart, Arm, Carrier)$

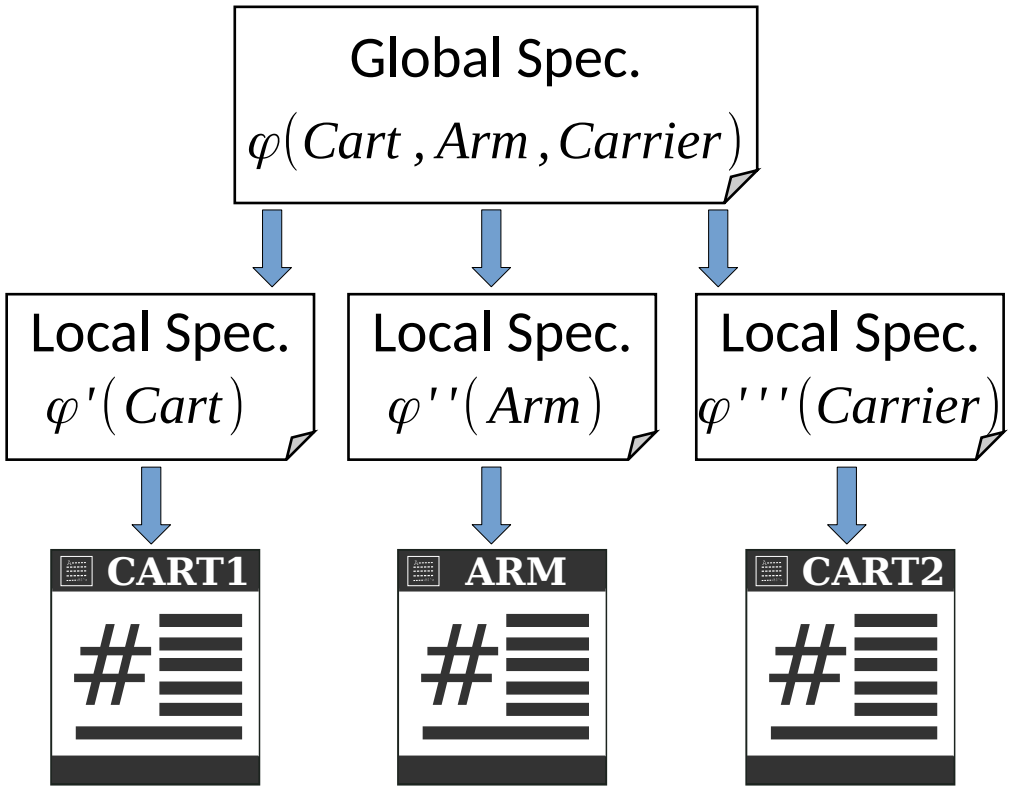
refines



The product is expensive.
(state-space explosion!)

We need to be compositional.

Compositional Verification with Sessions



Motion session type (Global)

Projections

Motion session type (Local)

Typing

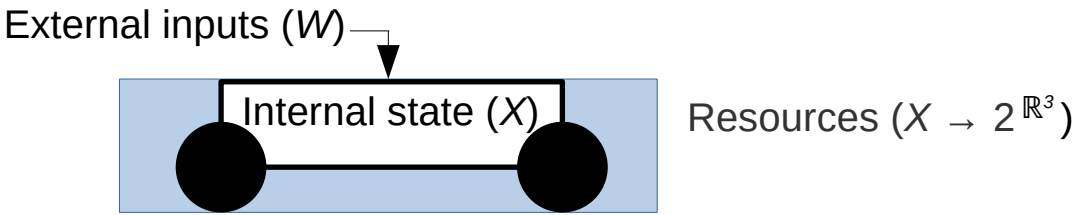
Implementation

In the Rest of this Talk

- Model, global and local types
- Projection
- (Sub)Typing
- Experiments
- Future

Model

- Communication: point-to-point, synchronous
- Computation: 0-time
- Physical components:



- Motion primitives (encapsulate low-level controller)
  - Fixed duration T (discrete time)
  - Pre/post conditions ($\mathbb{R}^X \times \mathbb{R}^W$), invariant ($[0;T] \rightarrow \mathbb{R}^X \times \mathbb{R}^W$)

Global Types

$$
\begin{aligned}
G \quad ::= \quad & \mathtt{dt}\langle (p_i : a_i) \rangle . G \\
\mid \quad & p \to q : \{\ell_i(S_i).G_i\}_{i \in I} \\
\mid \quad & \mu \mathbf{t}.G \quad \mid \quad \mathbf{t} \\
\mid \quad & \mathtt{end}
\end{aligned}
$$

Motion for each process

Communication & branching
(same sender+receiver for all choices)

Recursion
(no argument → regular language)

Global descriptions of the messages and the motions happening in the robots.

Local Types

$$
\begin{array}{rcl}
T & ::= & \mathsf{dt}\langle a \rangle.T \\
& \mid & \&\{p?\ell_i(S_i).T_i\}_{i \in I} \\
& \mid & \&\{p?\ell_i(S_i).T_i\}_{i \in I} \ \&\ \mathsf{dt}\langle a \rangle.T \\
& \mid & \oplus\{q!\ell_i(S_i).T_i\}_{i \in I} \\
& \mid & \mu \mathbf{t}.T \quad \mid \quad \mathbf{t} \\
& \mid & \mathtt{end}
\end{array}
$$

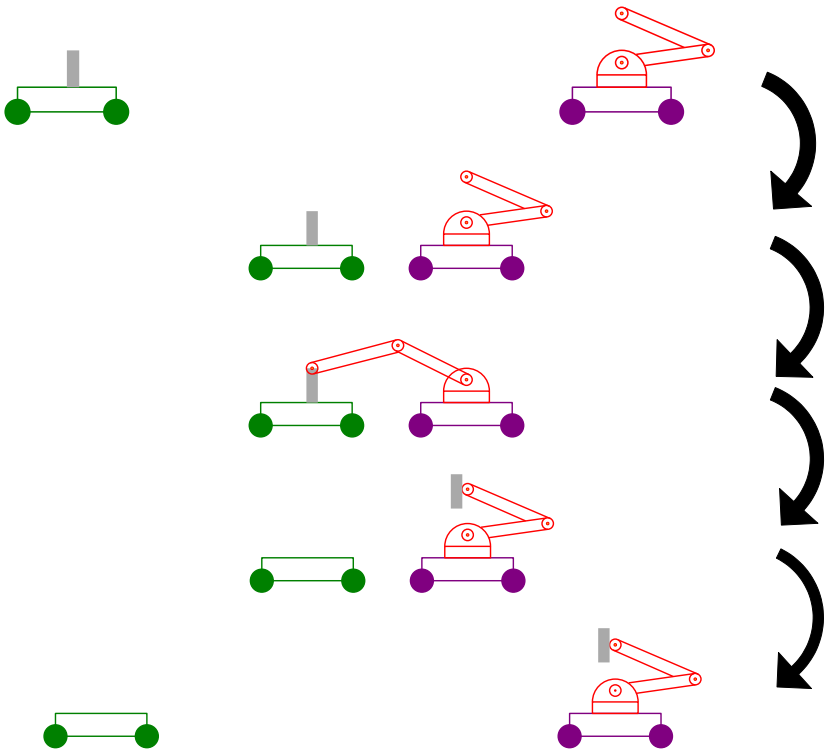
Motion

External choice
(receive + optional motion)

Internal choice + send

No mixed (internal+external) choice.
No send/receive with more than one other process.
Time is like a message broadcast to every process.

Handover: Global Type



$$\mathsf{Cart} \to \mathsf{Arm} : \mathit{fold}(\mathtt{unit}).\mathsf{dt}\langle \mathsf{Cart} : \mathsf{idle}, \mathsf{Carrier} : \mathsf{idle}, \mathsf{Arm} : \mathsf{fold}\rangle.$$

$$\mathsf{Arm} \to \mathsf{Cart} : \mathit{ok}(\mathtt{unit}).\mathsf{Cart} \to \mathsf{Carrier} : \mathit{ok}(\mathtt{unit}).$$

$$\mathsf{dt}\langle \mathsf{Cart} : \mathsf{move}, \mathsf{Carrier} : \mathsf{move}, \mathsf{Arm} : \mathsf{idle}\rangle.$$

$$\mathsf{Carrier} \to \mathsf{Cart} : \mathit{ok}(\mathtt{unit}).\mathsf{Cart} \to \mathsf{Arm} : \mathit{grab}(\mathtt{unit}).$$

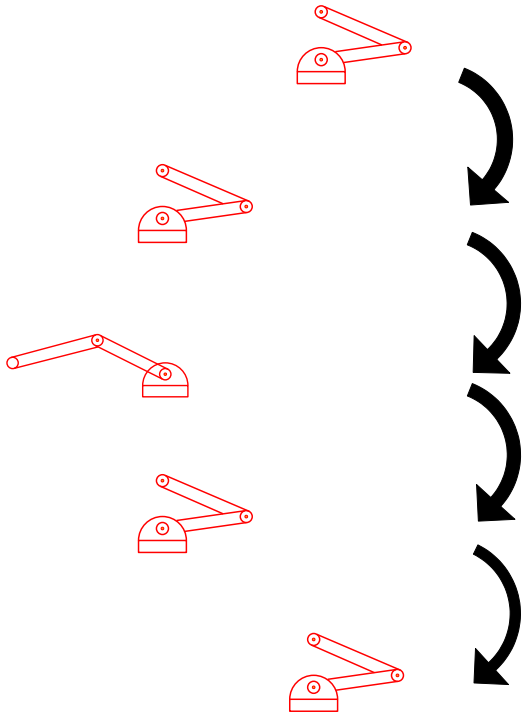
$$\mathsf{dt}\langle \mathsf{Cart} : \mathsf{idle}, \mathsf{Carrier} : \mathsf{idle}, \mathsf{Arm} : \mathsf{grip}\rangle.$$

$$\mathsf{Arm} \to \mathsf{Cart} : \mathit{ok}(\mathtt{unit}).\mathsf{Cart} \to \mathsf{Carrier} : \mathit{ok}(\mathtt{unit}).$$

$$\mathsf{dt}\langle \mathsf{Cart} : \mathsf{move}, \mathsf{Carrier} : \mathsf{move}, \mathsf{Arm} : \mathsf{idle}\rangle.$$

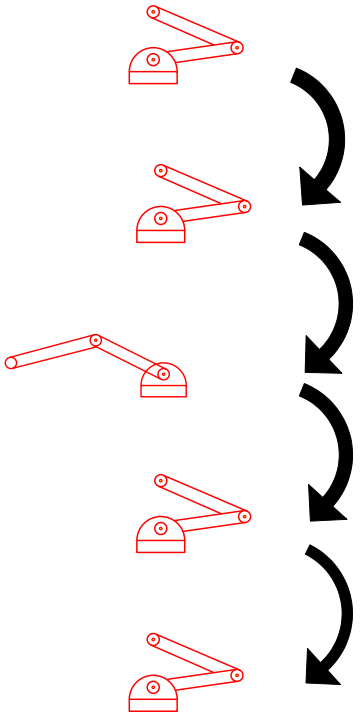
$$\mathsf{Cart} \to \mathsf{Arm} : \mathit{done}(\mathtt{unit}).\mathsf{Cart} \to \mathsf{Carrier} : \mathit{done}(\mathtt{unit}).\mathtt{end}$$

Handover: Projection on the Arm



$$\mathsf{Cart} \to \mathsf{Arm} : \mathit{fold}(\mathtt{unit}).\mathsf{dt}\langle\ \mathsf{Arm} : \mathsf{fold}\rangle.$$
$$\mathsf{Arm} \to \mathsf{Cart} : \mathit{ok}(\mathtt{unit}).$$
$$\mathsf{dt}\langle\ \mathsf{Arm} : \mathsf{idle}\rangle.$$
$$\mathsf{Cart} \to \mathsf{Arm} : \mathit{grab}(\mathtt{unit}).$$
$$\mathsf{dt}\langle\ \mathsf{Arm} : \mathsf{grip}\rangle.$$
$$\mathsf{Arm} \to \mathsf{Cart} : \mathit{ok}(\mathtt{unit}).$$
$$\mathsf{dt}\langle\ \mathsf{Arm} : \mathsf{idle}\rangle.$$
$$\mathsf{Cart} \to \mathsf{Arm} : \mathit{done}(\mathtt{unit}).\ \mathtt{end}$$

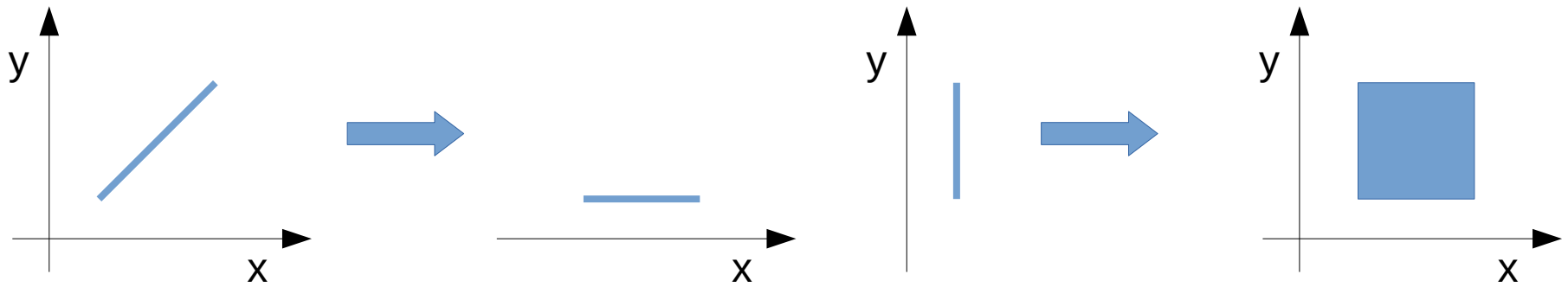
Handover: Projection on the Arm



$$
\begin{aligned}
&\mathsf{Cart}?\mathit{fold}(\mathtt{unit}).\\
&\qquad\qquad \mathsf{dt}\langle \mathsf{fold}\rangle.\\
&\qquad\qquad \mathsf{Cart}!\mathit{ok}(\mathtt{unit}).\\
&\mathsf{dt}\langle \mathsf{idle}\rangle.\\
&\mathsf{Cart}?\mathit{grab}(\mathtt{unit}).\\
&\qquad\qquad \mathsf{dt}\langle \mathsf{grip}\rangle.\\
&\qquad\qquad \mathsf{Cart}!\mathit{ok}(\mathtt{unit}).\\
&\mathsf{dt}\langle \mathsf{idle}\rangle.\\
&\mathsf{Cart}?\mathit{done}(\mathtt{unit}).\mathtt{end}
\end{aligned}
$$

Projection is Tricky

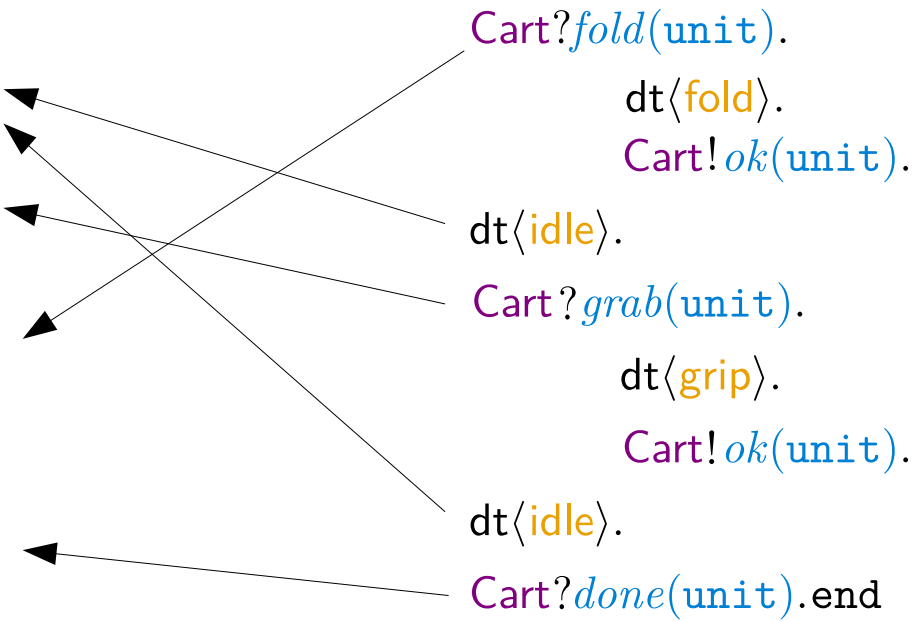
- “Projection then product” usually adds behaviors. (geometric analogy)



- Multiparty Session Type has a behavior preserving projection.
  - Only works on a restricted class of protocols.
  - Details in the paper.

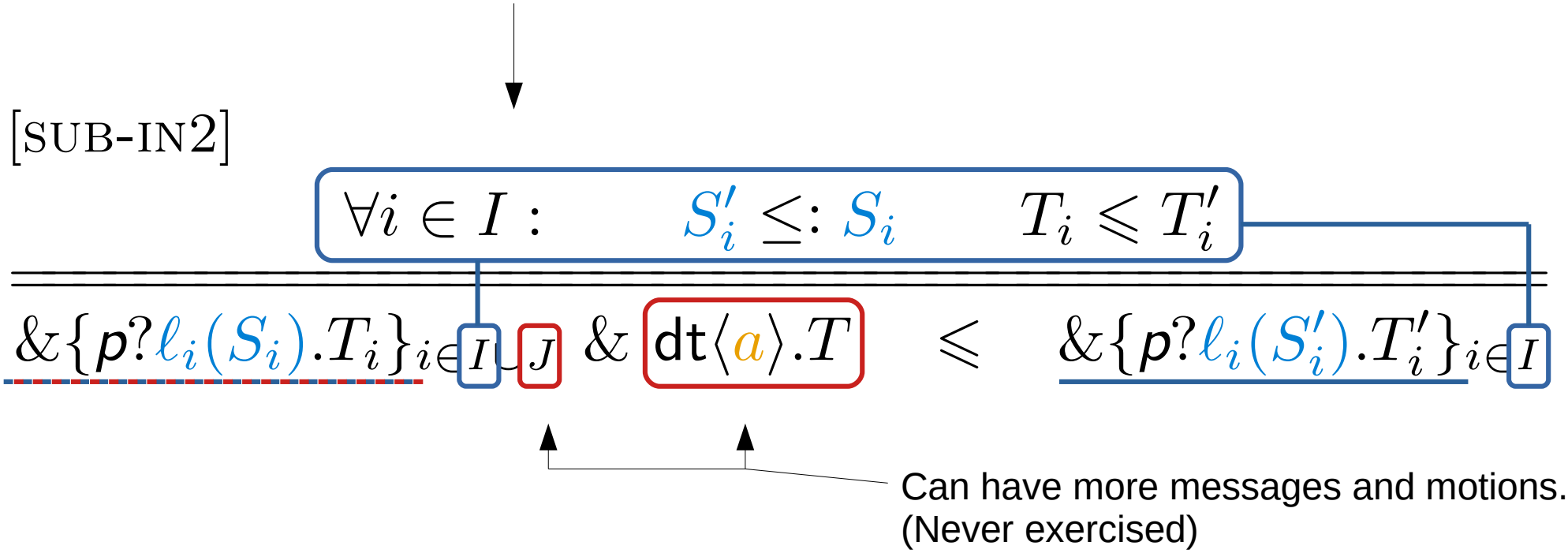
Typing the Arm: Not So Straightforward

```
while true do
   receive (Cart, idle)
      grab(loc) ⇒
         grip(loc);
         send(cart, ok)
      fold ⇒
         move(origin);
         send(cart, ok)
      done ⇒
         break
```



Subtyping: Receiving Messages

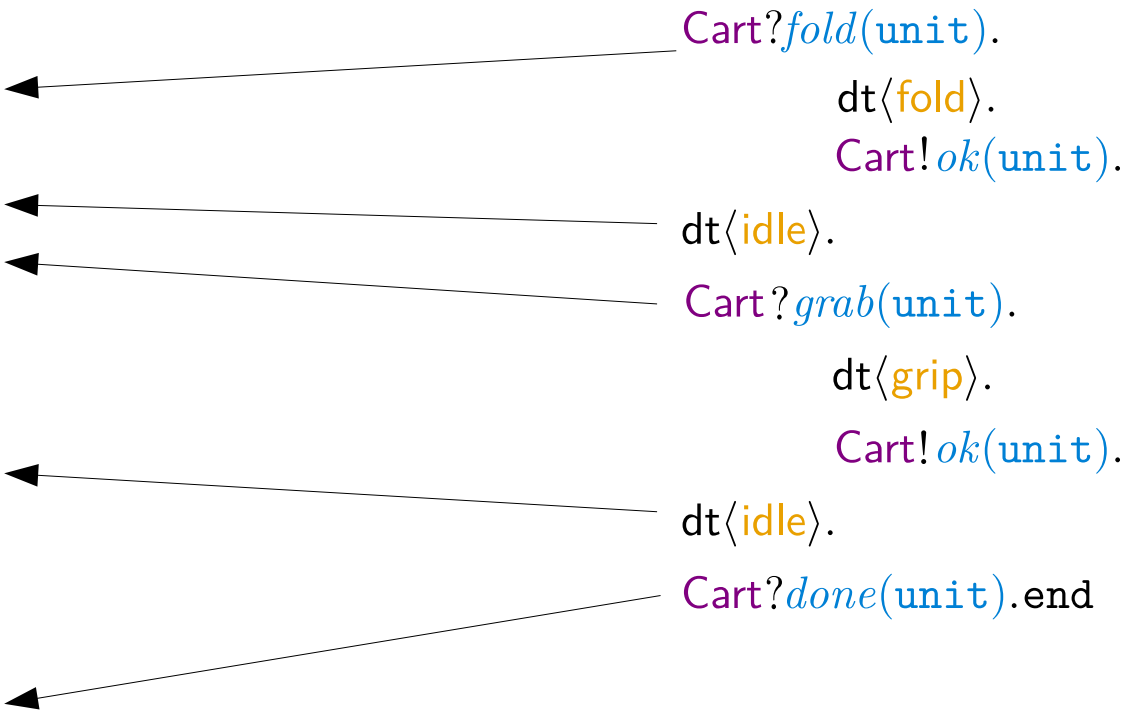
Common parts preserve subtyping.



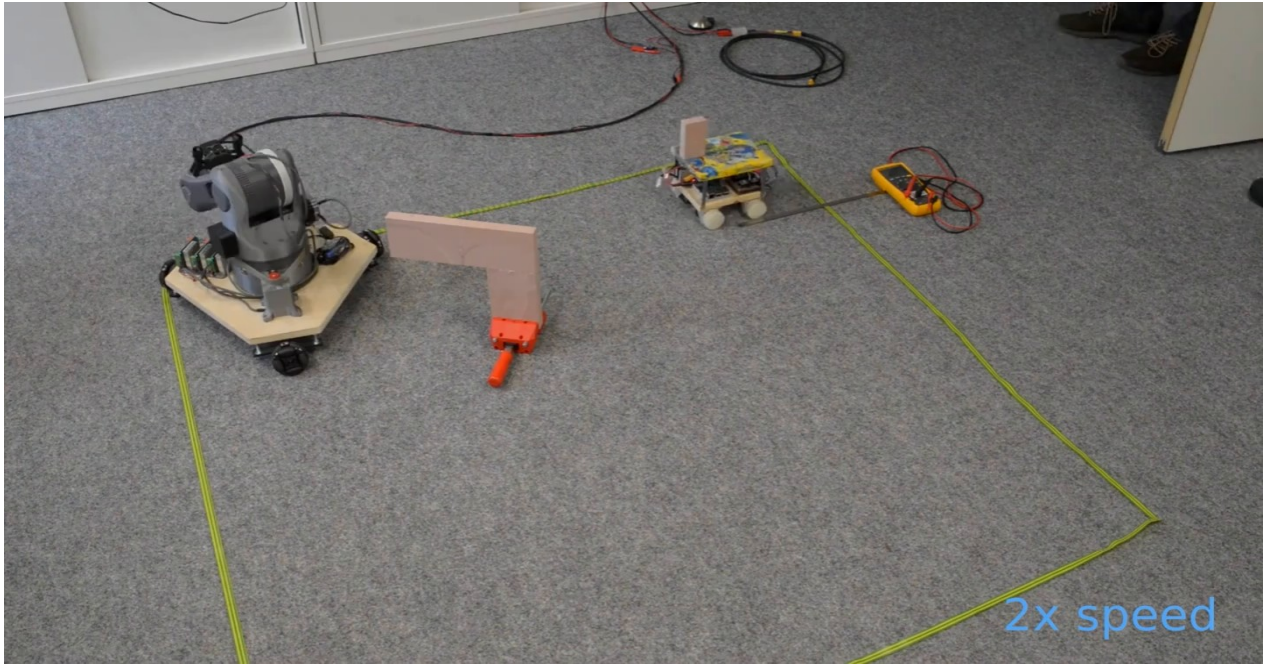
Using Subtyping

Loop unfolded 3 times:

```
receive (Cart, idle)
      grab(loc) ⇒
            grip(loc);
            send(cart, ok)
      fold ⇒
            move(origin);
            send(cart, ok)
      done ⇒
            break
receive (Cart, idle)
      grab(loc) ⇒
            grip(loc);
            send(cart, ok)
      fold ⇒
            move(origin);
            send(cart, ok)
      done ⇒
            break
receive (Cart, idle)
      grab(loc) ⇒
            grip(loc);
            send(cart, ok)
      fold ⇒
            move(origin);
            send(cart, ok)
      done ⇒
            break
```



Experiment: Underpass



Summary

- Synchronous multiparty session types + motion primitives
  - Session types: communication safety, deadlock-freedom
  - Motion primitives: integrates the physical world
  - Enable compositional verification
- Much more in the paper:

  Guarded choice, pre/post/inv, footprint and collisions, experiments, etc.
- Code: https://github.com/MPI-SWS/pgcd/

And Now?

- Reaching out, show the value outside of our community
- Part of a larger project:
  - [ICCPS 19]: Modularity and relative specifications
  - [*ECOOP 19*]: Session types with motion primitives
  - [IROS 19]: Motion primitive code and hardware
  - [under submission]: Unified proof system for motion primitives implementation and sessions
- Brave or foolish, time will tell...