Interfaces between Software Components for Robotic Manipulation

Stefaan C. Sonck Thiebaut
Interfaces for Robotic Manipulation

- General specification of force/motion tasks:
  Extension of kinematic task description,
  Control,
  Multi-contact/multi-robot experiment.

- Building the system from reusable components:
  Hierarchical data flow diagrams,
  Multiplicity and compositional interfaces,
  Reasoning with Semantic Properties.
General Specification of Force/Motion Tasks

- In "simple" cases, the Raibert/Craig model, used by Mason, works:
  - define a task frame
  - for every direction, specify whether it is
    - force-controlled or
    - motion-controlled

- Selection matrix $S$

- In more complex cases, that model fails, for example:
  - Solution: calculate the selection matrix, in an arbitrary frame as a function of the kinematic constraints $N$:
    
    $S(N)$

Work in collaboration with Prof. Oussama Khatib and Prof. Roy Featherstone
Some Experimental Results

- Force 1
- Force 2
Experiment
Interfaces for Robotic Manipulation

- General specification of force/motion tasks: Extension of kinematic task description, Control, Multi-contact/multi-robot experiment.

- Building the system from reusable components: Hierarchical data flow diagrams, Multiplicity and compositional interfaces, Reasoning with Semantic Properties.
The world is changing quickly

Many component technologies are under control

Fundamental Complexity in "Putting it all together"

Promise: fast deployment of powerful robotic systems
Reconfigurable - Reusable - Reliable
Using "ControlShell"

Finite-State Machines

Data-Flow Diagrams

Example of a Data-Flow Diagram:

\[ \frac{1}{1+\tau s} \]

Data-Flow Components (implemented in C++)

Data (Matrices)

\[ K_p \times (F-F_d) \times J^T \]

Joint Motors
The hidden semantic properties

- Is $F_{\text{raw}}$ in Newton or not?
- I am assuming the sample speed is at least 100 Hz
- Is it J or its transpose?
- I do hope the input isn’t bigger than 0.1 Nm

\[
\frac{1}{1+\tau s} F_{\text{raw}} \rightarrow F = K_p \times (F - F_d) \times J \rightarrow \text{Joint Motors}
\]
Smarter Components (more R-R-R)

The input must not be bigger than 0.1 Nm

Note for future users: only re-use this block-diagram for sample-speeds over 100Hz

Reconfigures itself!
Increase reliability in design through

"Design by Contract"

Concepts date from Dijkstra and Hoare (‘60s)
Combined with Object-Oriented Programming in:
- "Eiffel" language (Meyer, ’92);
- "Business Object Notation" (Walden & Nerson, ’95);
- "Toward Software Plug-and-Play" (Anderson Consulting, ’97);
- Several new development environments.

Pre-Conditions
what must be satisfied before this function can be called

Post-Conditions
what will be satisfied after this function has been called

example: \((x \neq 0)\) \(\Rightarrow y = f(x)\) \(\Rightarrow y = 1/x\)
Reliability in re-use is crucial!

Explosion of Arianne V in 1996

Some of the software developed for Arianne IV was reused incorrectly.
An example...

- **GripperGeometry**
- **WristSensor**
  - graspTransformation
  - force_in_SensorFrame
- **ForceTransformation**
  - force_in_GraspFrame
Example of an informal contract

- forceOut will be expressed in the same units as forceIn.
- forceIn has to be expressed in a frame consistent with the transformation.
- forceOut will be expressed in a frame consistent with the transformation.
- the rotation matrix in transformation has to be orthonormal.
How to put the contracts to work?

- Traditionally, these contracts are used in the design...
  ...as a formal documentation tool and
  ...to generate run-time checks and exceptions.

- But: this is not very useful in robot control!

\[ x \neq 0 \]
(Contracts \land \text{Semantic Properties} \land \text{Robots}) \Rightarrow F1

Run-time checking of Semantics...

...often is impossible \quad \text{...real-world}
\quad e.g.: the accuracy of this sensor has to be better than 0.1 rad

...often is prohibitively expensive \quad \text{...real-time}
\quad e.g.: this matrix has to be orthonormal to within accuracy...

...leads to exceptions \quad \text{...critical}

Contracts & Semantic Properties & Robots

Contracts have to be checked as much as possible
at \text{design-time}, not at run-time.
To control semantic information underflow, we need automatic propagation of semantic properties.
(Contracts ∧ Semantic Properties ∧ Robots) ⇒ F3

Can’t we rely on static type-checking and inheritance?

Add(3x1 Vector, 3x1 Vector)
Add(4x1 Vector, 4x1 Vector)
Add(3x1 Vector With Units of Newton, 3x1 Vector With Units of Newton)

Semantic Explosion of Data-types and Methods

Contracts & Semantic Properties & Robots
↓
To control semantic explosion, we need design-time analysis rather than static type checking and inheritance.
The challenge

Is it possible to...

• represent the conditions for correct re-use of these software components?

• check automatically, at design-time, whether these conditions are satisfied?

• propagate the semantic properties needed to check these conditions?

• turn this concept into a design tool for components that are more R-R-R?
Architecture

MDL/UID

MDL

MDL/ROBOT

Multi-Contact Experiment
MDL and formal verification

- More expressive power
- Reasoning more tractable
- Faster reconfigurability
- Higher reliability

"LOGIC" ORCCAD
INRIA, France
E. Coste-Manière, K. Kapellos, B. Espiau et al.
- Formal Verification of "constraints on events"
- Underlying formal language: Esterel
- Applied to: mobile and underwater robots

CONTROL SHELL + MDL

CONTROL SHELL
MDL must be a hybrid reasoning system:
- Must be able to take **logical decisions**
- Must be able to deal with **numerical data, matrices, ...**

MDL must be able to deal with:
- *constraints* (pre-conditions)
- *assertions* (post-conditions)

MDL must be performant enough for design-time use

MDL must be **hierarchical**:
- in the way rules are generated
- in the entities and properties
in relation with other work

- Frame-Based Reasoning
  "KL-ONE", "LOOM", ...

- Modal Description Logic

- Modal Logic

- Constraint Propagation (CP)
The knowledge is expressed as slots in a frame.

E.g.:

Matrix A
- numberOfColumns: 3
- numberOfRows: 3
- orthoNormal: true

Every slot can be a frame in itself.

The units of element 2_1 of matrix A is Nm

Eq(A/Matrix::2_1/units, Units::Nm)
and Constraint Propagation

Find the set of all possible worlds that satisfy a given set of constraints.

- **Simple Equality**
  - \( \text{Eq}(/A/\text{numberOfRows}, /A/\text{numberOfColumns}) \)
  - \( \text{Eq}(/A/\text{numberOfRows}, 3) \)

- **General Equality**
  - \( \text{Equal}(/A/\text{symmetric}, /A/\text{numberOfRows}, /A/\text{numberOfColumns}) \)
  - \( \text{Eq}(/A/\text{numberOfRows}, 2) \)
  - \( \text{Eq}(/A/\text{numberOfColumns}, 3) \)

\( \text{Eq}(/A/\text{symmetric}, \text{false}) \)
MDL has special rules enabling...

- ...logical decision making,
  
  $\text{Exor}(\text{.true.}, a, b)$

- ...numerical constraints,
  
  $\text{IntAddition}(\text{sum, term1, term2, term3})$

- ...matrix operations.
  
  $\text{ForAllIntegers}(i, 1, A/\text{numberOfElements})$
  
  \[
  \text{Eq}(A/\%i_1/\text{Units}, \text{Units::kg})
  \]
"Matrix A must be orthonormal"

```
Eq(/A/orthonormal?, true)
```

Something inadmissible might happen!

Possible Worlds
- /A/orthonormal=false
- /A/orthonormal=unknown

Admissible Worlds
- /A/orthonormal=true
"Matrix $A$ is guaranteed to be orthonormal"

\[ \text{Eq}([A/\text{orthonormal}], \text{true}) \]

Now, all possible worlds are admissible.
handles Units and Dimensions

Dimension {  
  DefinedBy([\text{length, mass, time}])  
  Type(length, ConstInt)  
  Type(mass, ConstInt)  
  Type(time, ConstInt) 
}

Dimension(Dimension::Force)  
Eq(Dimension::Force/length, 1)  
Eq(Dimension::Force/mass, 1)  
Eq(Dimension::Force/time, -2)

Units {  
  DefinedBy([\text{dimension, scale}])  
  Dimension(dimension)  
  Type(scale, ConstReal) 
}

Units(Units::cm, "centimeter")  
Eq(Units::cm/dimension,  
  Dimension::Length)  
Eq(Units::cm/scale, 0.01)

Dimension ≡  
\begin{align*}  
\text{length} & \quad \text{mass} & \quad \text{time} \\
\text{Length} & \quad \times & \quad \text{Mass} & \quad \times & \quad \text{Time} \\
\end{align*}

Units ≡  
\begin{align*}  
\text{scale} & \quad \times & \quad \text{m} & \quad \times & \quad \text{kg} & \quad \times & \quad \text{s} \\
\end{align*}

Units::cm ≡  
0.01 \times \text{m}
handles Frames

...  
Eq(f_out/force!, f_in/force?)

Exor(true, noInvert, invert)
Type(noInvert, ConstBool)
Type(invert, ConstBool)

Eq(noInvert, trsf/fromBase?, v_in/inBase?)
Eq(noInvert, trsf/toBase?, v_out/inBase!)
Eq(invert, trsf/toBase?, v_in/inBase?)
Eq(invert, trsf/fromBase?, v_out/inBase!)
...

This component can Reconfigure itself!
The Design-Cycle

MDL User-Interface

Data-Flow User-Interface

Automatically Generated MDL

Generate a run-time system (in ControlShell)
Problem of hierarchies

\[ Eq(x/\text{Units}, y/\text{Units}) \]

\[ x \]

\[ y \]

compA

compB

compC

\[ Eq(\text{compC/compB/compA/x/Units}, \text{compC/compB/compA/y/Units}) \]

X

Y

\[ Eq(\text{X/Units}, \text{Y/Units}) \]
How to name things in the world?

Calculate the lowest cost path to the thing you want to refer to.

Some examples:

\[ /\text{DistanceVector}/1\_1/\text{Units}::\text{units} \]

\[ \text{cost} = 10 + 10 + 1 = 21 \]

\[ /\text{Units}::\text{km} \]

\[ \text{cost} = 1 \]

Properties are organized hierarchically and have a cost associated with them.

\[ /\text{Force}/\text{Matrix}::\text{numberOfRows} \]

\[ /\text{ForceBase}/\text{Frame}::\text{numberOfDegreesOfFreedom} \]

Here it depends on whether the user is more interested in properties about matrices or frames.
Example: an error in units occurred

These two things cannot be equal:

Units::N
Units::Nm

MDL rules involved in this problem:
- organized hierarchically (in a tree)
- icons indicate whether a rule is satisfied, pending or violated
- what line in what file of MDL-code the rule originates from
Example: Not enough information!

Violation!

This rule is defined (in the library) as follows:

MatrixWithUnits(/A)

Give hints to solve the problem

"have to know the precise number of columns in the matrix"
Example: Reasoning with frames

- had to add information about the worldframe
- had to add extra information to the transformation between the worldframe and the robotframe
- automatically configured about 10 invert/not-invert components inside this component
- information about the meaning of all these transformations
- checked approximately 25 units (in this connection alone)
Summary: contributions

- First demonstration of a fine-motion task with multiple contacts: Generalized kinematic task-description for fine-motion.
- Framework for R-R-R software-components for robotics:
  - Identified experimentally the requirements F1...F3;
  - Defined the Modal Description Logic;
  - Designed an efficient reasoning engine for MDL;
  - Looked at issues of complexity in user interaction;
  - Used MDL to encode common knowledge in Robotics;
  - Verified this experimentally.
What lies ahead?

- Expansion of this approach to other domains
  - Reasoning about time;
  - Finite State Machines
    (but other tools might be more appropriate...)
  - More extensive reasoning about "accuracy"

- Careful definition of vocabulary

  A/orthonormal ≠ A/OrthoNormality

  A/accuracy

  But: do they mean the same thing?

Need an "ontology"