Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Maude Object-Oriented Action Tool

André Murbach Maidl[1]   Cláudio Carvilhe[2]
Martin A. Musicante[3]

[1]Universidade Federal do Paraná

[2]Pontifícia Universidade Católica do Paraná

[3]Universidade Federal do Rio Grande do Norte

Logical and Semantic Frameworks, with Applications 2007

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Introduction

Introduction

- ► AS is a formal framework.
- ► Its notation presents good reusability and extensibility properties.
- ► However, AS lacks of syntactic support for the definition of modules and libraries.
- ► A modular approach introduces modules into AS descriptions.
- ► Based on the modular approach and object-orientation, OOAS was proposed.

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Introduction

Introduction

- ▶ OOAS was defined using SOS transitions.
- ▶ No tool for OOAS had been implemented yet.
- ▶ Here we present MOOAT:
  - ▶ the first OOAS implementation;
  - ▶ changes on OOAS syntax;
  - ▶ OOAS implementation using MSOS;
  - ▶ how the tool has been built using Full Maude and Maude MSOS Tool.
- ▶ In addition, we show how to use Mosses' *constructive* approach with OOAS as a case study of MOOAT.

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Maude MSOS Tool
MMT limitations

Maude MSOS Tool

- ▶ It is an executable environment for describing MSOS specifications.
- ▶ It is a conservative extension of Full Maude that compiles MSOS specifications into Rewriting Logic.
- ▶ Uses an adopted MSDF syntax:
  - ▶ Modular SOS Definition Formalism - Mosses.
- ▶ MMT is quite useful:
  - ▶ the abstract syntax is defined using BNF;
  - ▶ the semantics is given by *labeled transitions*, which contains the necessary semantic components;
  - ▶ a specification might be organized by modules.

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Maude MSOS Tool
MMT limitations

MMT limitations

▶ *pre-regularity check*
  ▶ A term must contain just one sort.

```
Wrong: Storable ::= Int . Value ::= Int .

Correct: Storable ::= Int . Value ::= Storable .
```

▶ *ad-hoc overloading*
  ▶ If the sorts in the arities of two operators with the same syntactic form are pairwise in the same connected components, then the sorts in the coarities must likewise be in the same connected component.

```
Wrong: Exp ::= sum (Exp, Exp) | Int | Id . Env = (Id, Int) Map . Sto ::= (Int, Int) Map .

Correct: Exp ::= sum (Exp, Exp) | < Int > | Id . Env = (Id, Int) Map . Sto ::= (Int, Int) Map .
```

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Object-Oriented Action Semantics
OOAS example
State class

Object-Oriented Action Semantics

▶ Method to organize AS specifications.
▶ Goals:
  ▶ enhance extensibility and reusability;
  ▶ split semantic descriptions into classes;
  ▶ treat semantic functions as methods.
▶ OOAS extends AN:
  ▶ to accept classes and operators based on object-orientation;
  ▶ a class is splited into two parts: syntax and semantics;
  ▶ an OOAS description is a hierarchy of classes.

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Object-Oriented Action Semantics
OOAS example
State class

OOAS example

Class Command
    syntax:
        Cmd
    semantics:
        execute _ : Cmd $\rightarrow$ Action
End Class

Class While
    extending Command
    using E:Expression, C:Command
    syntax:
        Cmd ::= "while" E "do" C
    semantics:
        execute ⟦ "while" E "do" C ⟧ =
                unfolding
                 | evaluate E then
                 | | execute C and then unfold else complete
End Class

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Object-Oriented Action Semantics
OOAS example
State class

State class

- ▶ It is the base-class for all OOAS classes.
- ▶ Since all classes are under a pre-defined root class, they belong to a hierarchy of classes.
- ▶ It is responsible to implement the attributes and operations that are used in those classes specifications.
- ▶ OOAS actions are exactly the same actions of AS.
- ▶ *State* attributes:
  - ▶ *transients*;
  - ▶ *bindings*;
  - ▶ *storage*.

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Object-Oriented Action Semantics
OOAS example
State class

State class

- ▶ Some operations are available to handle *State* attributes.
- ▶ They are classified according to AS facets:
    - ▶ basic - control flow;
    - ▶ functional - transients;
    - ▶ declarative - bindings;
    - ▶ imperative - storage;
    - ▶ reflective - abstractions;
    - ▶ hybrid - actions that deal with more than one facet.
- ▶ No operation was defined to deal with communicative facet.
- ▶ The formal semantics of each operation has been defined using SOS.

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Constructive Action Semantics
Constructs
CAS example

Constructive Action Semantics

▶ Constructs are defined separately and they can be used in
  several specifications by deriving constructs.

▶ The set of constructs is not minimal, in a specification just
  the necessary constructs are included in it.

▶ In this way, all available constructs do not need to be included
  in every definition.

▶ The constructive approach provides a huge flexibility on the
  definition of programming languages.
  ▶ In this regard, it has been used with Modular Action Semantics
    and Modular Structural Operational Semantics.

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Constructive Action Semantics
Constructs
CAS example

Constructs

- ▶ A construct is the formal specification of a certain programming language feature.
- ▶ Abstract and Concrete Constructs:
  - ▶ while (Exp) do Cmd (concrete);
  - ▶ cond-loop(Exp, Cmd) (abstract);
  - ▶ a language-independent prefix notation is used to define abstract constructs.
- ▶ Basic and Derived Constructs:
  - ▶ basic constructs represent common features that have the same interpretation and are found in many programming languages;
  - ▶ derived constructs represent particular features regarding the programming language;
  - ▶ derived constructs are obtained by combining abstract constructs.

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Constructive Action Semantics
Constructs
CAS example

CAS example

Module Cmd
requires C:Cmd
semantics execute : Cmd → Action

Module Cmd/While
syntax Cmd ::= cond-loop(Exp, Cmd)
requires Val ::= Boolean
semantics execute cond-loop(E, C) =
    unfolding
        evaluate E then
            execute C and then unfold else complete

▶ The CAS goal is to map concrete constructs from
  programming languages to a set of basic abstract constructs.

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Maude Object-Oriented Action Tool
MOOAT example
Data Notation
Action Notation
State class
Classes Notation

Maude Object-Oriented Action Tool

- ▶ MOOAT
  - ▶ The first executable environment for describing programming languages using OOAS.
  - ▶ MMT has been used to specify the Action Notation.
  - ▶ Full Maude has been used to implement the Classes Notation.
  - ▶ MOOAT development was inspired by MAT (Maude Action Tool).
- ▶ MOOAT is a conservative extension of Full Maude and MMT
  - ▶ It has the same limitations as Full Maude and MMT.
  - ▶ Regarding these limitations, its syntax has some differences if compared to the formalism one.
  - ▶ Yet, MOOAT syntax and OOAS syntax are quite similar.

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Maude Object-Oriented Action Tool
MOOAT example
Data Notation
Action Notation
State class
Classes Notation

MOOAT example

```
(class Command is
 Cmd .
 absmethod execute Cmd -> Action .
endclass)

(class While is
 extends Command .
 Cmd ::= while Exp do Cmd .
 method execute (while E:Exp do C:Cmd) = unfolding
    ((evaluate E:Exp) then
    ((execute C:Cmd and then unfold) else complete)) .
endclass)
```

▶ An OOAS specification is a finite set of classes. The classes create the
necessary hierarchy. Syntax trees in MMT BNF style are used. The
semantics is given by Action Notation.

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Maude Object-Oriented Action Tool
MOOAT example
Data Notation
Action Notation
State class
Classes Notation

Data Notation

- ▶ DN has been implemented using 1 MSDF module and 1 Maude system module.
    - ▶ MSDF - DataNotation
        - ▶ The sorts of data used by AN are defined.
        - ▶ Action, Yielder and Data - *sorts* implemented by MOOAT.
        - ▶ Int and Boolean - *sorts* implemented by MMT and used by MOOAT.
    - ▶ Maude - DATANOTATION
        - ▶ Implement the function op _:<_ : Data DataSort -> Bool.
        - ▶ Used to verify if a datum belongs to a *sort*.
        - ▶ < 1 > :< value returns true.
        - ▶ < cell (1) > :< abstraction returns false.
    - ▶ Some auxiliary operations has been defined:
        - ▶ sum (Yielder, Yielder).

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Maude Object-Oriented Action Tool
MOOAT example
Data Notation
Action Notation
State class
Classes Notation

Action Notation

- ▶ AN has been splited into MSDF modules:
    - ▶ the modules has been used to implement the available facets;
    - ▶ AN specifies the indexes used in the transition labels.

```
(msos BasicSyntax is
 see BasicData .

 Action ::= Action or Action | fail | commit |
            Action and Action | complete |
            indivisibly Action |
            Action and'then Action |
            Action trap Action | escape |
            unfolding Action | unfold | diverge .


 Yielder ::= the DataSort yielded'by Yielder |
             nothing | Data .
sosm)
```

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Maude Object-Oriented Action Tool
MOOAT example
Data Notation
Action Notation
State class
Classes Notation

Action Notation

```
(msos FunctionalOutcomes is
 see BasicOutcomes .
 see FunctionalData .

 Completed .

 Terminated ::= Completed .

 Completed ::= completed .
 Completed ::= gave (Data) .

 gave (none) : Action --> completed .
sosm)
```

```
(msos BasicConfigurations is
 see BasicSyntax .
 see BasicOutcomes .

 Action ::= Terminated |
            Action @ Action .
sosm)
```

```
(msos FunctionalLabels is
 see BasicLabels .
 see FunctionalData .

 Label = {data : Transients,
          data' : Transients, ...} .
sosm)
```

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Maude Object-Oriented Action Tool
MOOAT example
Data Notation
Action Notation
State class
Classes Notation

Action Notation

- Each facet implements the necessary transitions.
- We have basically 3 kinds of transitions:
  - those transitions that neither change or use labels' components;

    ```
                Action1 -{...}-> Action'1
    -- ---------------------------------------------------------
    Action1 and Action2 : Action -{...}-> Action'1 and Action2 .

                Action2 -{...}-> Action'2
    -- ---------------------------------------------------------
    Action1 and Action2 : Action -{...}-> Action1 and Action'2 .
    ```

  - those transitions that just use labels' components;

    ```
    regive : Action -{data = Transients, data' = Transients,-}-> gave (< Transients >) .
    ```

  - those transitions that either change and use label's components;

    ```
            Transients' := (1 |-> Data1) / Transients,
            Transients'' := (2 |-> Data2) / Transients'
    -- -------------------------------------------------------------------------------
    gave (Data1) and gave (Data2) : Action -{data = Transients, data' = Transients'',-}->
        gave (concatenation(Data1, Data2)) .
    ```

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Maude Object-Oriented Action Tool
MOOAT example
Data Notation
Action Notation
State class
Classes Notation

State class

- ▶ It is represented by a Maude system module called STATE.
  - ▶ Since MOOAT classes are translated to system modules and also because all MOOAT classes are subclasses of STATE as well as in OOAS.
- ▶ And also by Classes Notation.
- ▶ MOOAT *State* class has 5 attributes.

```
(mod STATE is
 including ActionNotation .
 op init-rec : -> Record .
 eq init-rec = {commitment = false,
                unfolding = fail,
                data = (void).Map{Int,Datum},
                bindings = (void).Map{Token,Data},
                storage = (void).Map{Cell,Data}} .
 var A : Action .
 var S : Storage .
 eq new-cell (S) = cell (length (S) + 1) .
 op compute : Action -> Conf .
 eq compute (A) = < A ::: 'Action, init-rec > .
endm)
```

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Maude Object-Oriented Action Tool
MOOAT example
Data Notation
Action Notation
State class
Classes Notation

Classes Notation

- ▶ CN has been developed similar to:
    - ▶ Full Maude object-oriented modules;
    - ▶ MMT MSDF modules.
- ▶ When CN syntax is used its constructs are translated to code that Maude is able to interpret.
- ▶ MOOAT classes are composed basically by 3 parts:
    - ▶ extends part;
    - ▶ syntactic part;
    - ▶ semantics part.
- ▶ The use of these parts might be optional or sequential. However, a class must has at least one of these 3 parts.

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Maude Object-Oriented Action Tool
MOOAT example
Data Notation
Action Notation
State class
Classes Notation

Classes Notation

- ▶ Each class part is translated as follows:
  - ▶ subclasses definitions are simply translated to lines that use Maude's directive including;
  - ▶ syntactic definitions use the BNF style introduced by MMT - trees and syntactic sorts are translated to Maude's operations, sorts and subsorts;
  - ▶ in the semantic definitions, methods and objects are translated respectively to Maude's equation sets and meta-variables.
- ▶ A methods environment is created by the inclusion of the converted MOOAT classes into Maude's module database.

```
rl < O : X@Database | db : DB, input : ('class_is_endclass[T, T']), step-flag : B,
                      output : nil, default : MN, Atts > =>
   < O : X@Database | db : mooat-proc-unit('class_is_endclass[T, T'], step-flag(B), DB),
                      input : nilTermList, step-flag : B, output : ('Introduced 'OOAS 'class
                      header2QidList(parseHeader(T)) '\n), default : parseHeader(T), Atts > .
```

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Constructive Object-Oriented Action Semantics
μ-Pascal using COOAS

Constructive Object-Oriented Action Semantics

- COOAS = CAS + OOAS.
- Case study of MOOAT.

```
(class Cmd is
 Cmd .
 absmethod execute Cmd -> Action .
endclass)

(class Cmd/While is
 extends Cmd .
 Cmd ::= cmd-while (Exp, Cmd) .
 method execute (cmd-while(E:Exp, C:Cmd)) = unfolding
     (evaluate E:Exp then
     ((execute C:Cmd and then unfold) else complete)) .
endclass)
```

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Constructive Object-Oriented Action Semantics
μ-Pascal using COOAS

μ-Pascal using COOAS

- ▶ μ-Pascal is a toy language fairly similar to the imperative language Pascal.
- ▶ It contains basically commands and expressions.
- ▶ It is being described just by extending the necessary COOAS classes to compose the language.

```
(class Micro-Pascal is
 extends Exp/Val, Exp/Val-Id .
 extends Exp/Sum, Exp/Sub, Exp/Prod .
 extends Exp/True, Exp/False, Exp/LessThan, Exp/Equality .
 extends Cmd/Assignment, Cmd/Repeat, Cmd/While .
 extends Cmd/Sequence, Cmd/Cond .
 extends Dec/Variable, Dec/DecSeq .
 extends Prog .
endclass)
```

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Final Remarks and Future Work
Thank you

Final Remarks and Future Work

- ▶ We have presented MOOAT, the first OOAS implementation.
- ▶ MOOAT is a conservative extension of Full Maude and MMT.
- ▶ The previous OOAS specification using SOS has been rewritten using MSOS and implemented using MMT.
- ▶ A language to specify OOAS classes has been built in Maude.
- ▶ MOOAT is the update of OOAS since its formal specification has been translated from SOS to MSOS.
- ▶ MOOAT notation covers a rich set of actions and action combinators, which includes the complete OOAS notation.

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Final Remarks and Future Work
Thank you

Final Remarks and Future Work

- ► COOAS represents a novel view to the OOAS system.
- ► This is the first time that CAS and OOAS are combined.
- ► The modularity aspects observed in OOAS has been improved due the introduction of constructs into it.
- ► Since such combination is capable to describe syntax-independent specifications of programming languages.
- ► As future work:
  - ► we would implement the communicative facet of Action Notation;
  - ► we would trace a careful comparison between COOAS and a library of OOAS classes called LFLv2.

Introduction
Maude MSOS Tool
Object-Oriented Action Semantics
Constructive Action Semantics
Maude Object-Oriented Action Tool
Constructive Object-Oriented Action Semantics
Final Remarks and Future Work

Final Remarks and Future Work
Thank you

Thank you

- ► MOOAT can be found at:
  - ► http://www.inf.ufpr.br/murbach/mooat/
- ► Please, send any questions or comments to:
  - ► André Murbach Maidl - murbach@inf.ufpr.br