# MAccS: a Tool for Reachability by Design

Guillaume Verdier, Jean-Baptiste Raclet

# MAccS: A Tool for Reachability by Design

Guillaume Verdier(✉) and Jean-Baptiste Raclet

IRIT/CNRS, 118 Route de Narbonne, 31062 Toulouse Cedex 9, France
{verdier,raclet}@irit.fr

**Abstract.** MAccS is a tool for the modular design of complex IT systems. Component specifications are given in the form of marked acceptance specifications which are acceptance specifications, an extension of modal specifications, enriched with reachability constraints on states. The tool supports the crucial operators for a complete specification theory: satisfaction checking, consistency, refinement, product, quotient and conjunction. These operators can be used to build larger systems by composing or decomposing component specifications while ensuring some reachability properties.

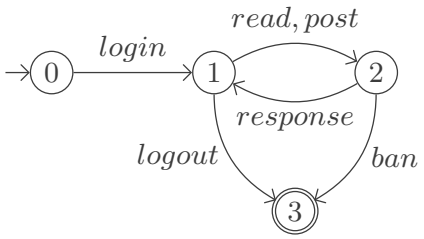
## 1 Introduction

The basic idea underlying modular design is to break down complex systems into individual components that can be implemented concurrently or possibly taken off-the-shelf, and later composed to obtain the targeted system. This approach can be supported by a specification theory in which a formalism is defined for the component specifications together with dedicated operators allowing to perform different steps of a system design flow.

The tool MAccS implements a specification theory based on *marked acceptance specifications* (MAS) [1]. As an example of MAS, consider the specification *Server* shown in Fig. 1. It consists of a finite deterministic transition system with, associated to any state $q$, a set $\mathrm{Acc}(q)$ of sets of actions that may be enabled in a model of the specification. In general terms, a MAS characterizes a (possibly infinite) set of finite transition systems called its *models*. Now Fig. 2 depicts two models of *Server* (for a formal definition of satisfaction, see [1]). In each state of the model, the set of outgoing transitions must be an element of the acceptance set of the specification. For example, the state 2 of the MAS *Server* allows either to do only one transition labeled *response* or two transitions labeled *response* and *ban*. In the model on the right of Fig. 2, the state 2 chose this last option (both *response* and *ban*) while the state 4 only realizes the transition *response*.

Observe that *logout* and *ban* are optional respectively in state 1 and in state 2 of *Server* as these actions are not present in all sets in $\mathrm{Acc}(1)$ and $\mathrm{Acc}(2)$ and thus may not be present in some models of the specification. Moreover, state 3 of *Server* is marked to encode the constraint that it must be reached in any

Source code of MAccS, screenshots and more examples are available at http://irit.fr/Guillaume.Verdier.




$\mathrm{Acc}(0) = \{\{login\}\}$
$\mathrm{Acc}(1) = \{\{read, post\}, \{read, post, logout\}\}$
$\mathrm{Acc}(2) = \{\{response\}, \{response, ban\}\}$
$\mathrm{Acc}(3) = \{\emptyset\}$

**Fig. 1.** A MAS *Server*

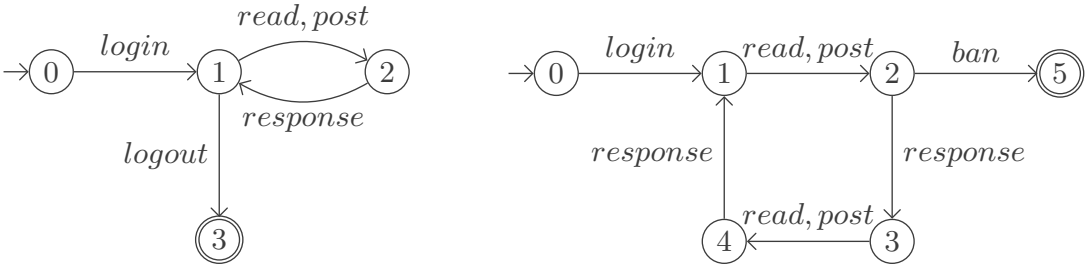



**Fig. 2.** Two models of the MAS *Server*

model. As a result, although the actions *logout* and *ban* are optional, at least one of the two must be present in any model of *Server*. This kind of constraint entails that MAS are more expressive than modal transition systems (MTS) [2]. In this case, marking the state 3 is used to express the termination of the service, as the marked state has no outgoing transition, but one may also mark some states with outgoing transitions to ensure the reachability of checkpoints or a liveness property.

Consequently, the MAS *Server* specifies the behavior of a forum-like site: after having logged in, one can read messages or post a new message and may eventually log out. The server may also eventually decide to ban users.

A specification theory must support a set of crucial operators to be regarded as complete [3]. This is the case for the one implemented in MAccS which supports satisfaction checking, consistency and refinement for substitutability. It also supports product for the composition of MAS, quotient for the decomposition of MAS and conjunction for the merge of viewpoints modeled as MAS. These operators preserve the reachability of marked states and thus guarantee by design some reachability properties.

## 2 The Tool MAccS

MAccS offers a Graphical User Interface (GUI) featuring an interactive view of transition systems and MAS allowing to edit them easily and a sidebar to select and apply different operations defined on MAS. MAccS is also available as a library for integration in other programs or automatic processing.

MAccS is written in standard C++; the graphs underlying the transition systems and MAS are handled by the Boost Graph Library [4]. The GUI is

```
init {{login}}
wait {{read,post},{read,post,logout}}
...
init -login-> wait
wait -read-> reply
reply -response-> wait
...
```

**Fig. 3.** Excerpt of the textual form of the MAS given in Fig. 1

made with the framework Qt and Dot [5] is used to generate the layout of the transition systems and MAS. MAccS does not make use of platform-specific libraries and should thus compile and run on most desktop operating systems.

In addition to being created and modified through the GUI, the transition systems and MAS may be written in a simple textual format. An excerpt of this format is shown in Fig. 3 which corresponds to the MAS *Server* in Fig. 1. It is also possible to import and export them from and to the Dot format [5].

After creating (or importing) some transition systems or some MAS, several operations are available that we now introduce briefly.

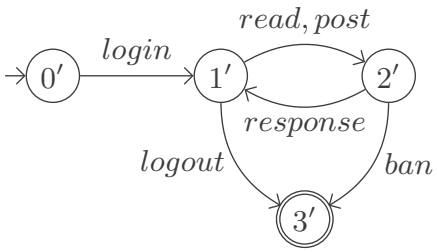
*Satisfaction checking.* As mentioned in the introduction, a MAS characterizes a set of transition systems. One may first test in MAccS if a transition system $M$ is a model of a given MAS $S$.

For example, it can be verified that the left-hand side transition system in Fig. 2 is a model of the MAS *Server* in Fig. 1. It corresponds to the implementation of *Server* in which users are never banned: after the *login* transition, the set $\{read, post, logout\}$ was selected from the Acc(1) and the set $\{response\}$ was chosen from Acc(2).

*Refinement of MAS.* Refinement allows one to replace, in any context, a specification by a more detailed version of it. Substitutability of a MAS $S_2$ by a MAS $S_1$ is allowed when every model satisfying the refinement $S_1$ also satisfies the larger specification $S_2$. Inclusion of the sets of models, also referred as thorough refinement in the literature, can be tested for two MAS in MAccS thus enabling to decide refinement.

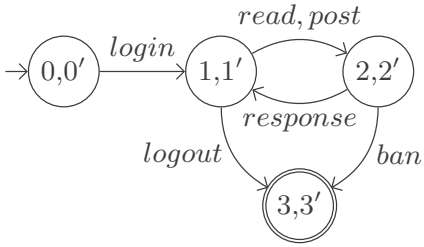
*Product of MAS.* Two MAS $S_1$ and $S_2$ may be composed using the *product* operation (denoted $\otimes$), which synchronizes their actions. However, reachability is not compositional in general meaning that there may exist some models $M_1$ of $S_1$ and $M_2$ of $S_2$ such that from some states of $M_1 \times M_2$ no pair of marked states can be reached.

In order to enable the concurrent implementation of MAS while guaranteeing some reachability constraints, we have proposed in [1] a *compatible reachability* criterion which is a precondition for the computation of $S_1 \otimes S_2$ and allows to check if a pair of marked states is always reachable in the product of any two models of the MAS.




$\mathrm{Acc}(0') = \{\{login\}\}$
$\mathrm{Acc}(1') = \{\{read, post, logout\}\}$
$\mathrm{Acc}(2') = \{\{response, ban\}\}$
$\mathrm{Acc}(3') = \{\emptyset\}$

**Fig. 4.** MAS *Client*




$\mathrm{Acc}(0, 0') = \{\{login\}\}$
$\mathrm{Acc}(1, 1') = \{\{read, post\}, \{read, post, logout\}\}$
$\mathrm{Acc}(2, 2') = \{\{response\}, \{response, ban\}\}$
$\mathrm{Acc}(3, 3') = \{\emptyset\}$

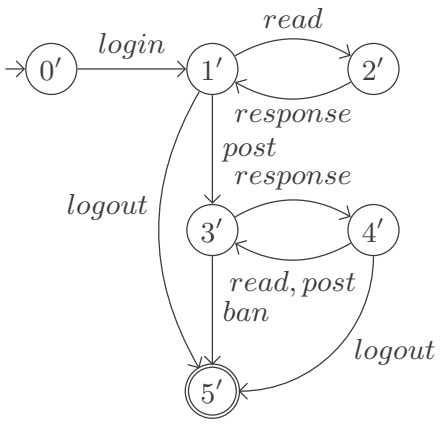
**Fig. 5.** Result of $Server \otimes Client$

For example, the MAS *Server* given in Fig. 1 and *Client* given in Fig. 4 have a compatible reachability and their product is shown in Fig. 5.

When some MAS $S_1$ and $S_2$ do not have a compatible reachability, MAccS proposes to compute the largest refinement $S_2'$ of $S_2$ such that $S_1$ and $S_2'$ have a compatible reachability.

*Quotient of MAS.* Conversely, we may decompose specifications with the *quotient* operation. Intuitively, it is the opposite of the product: given two MAS $S$ and $S_1$, the MAS $S/S_1$ is such that any of its models composed with any model of $S_1$ is a model of $S$. It also allows component reuse as $S_1$ may be typically the specification associated to a grey box component available off-the-shelf.

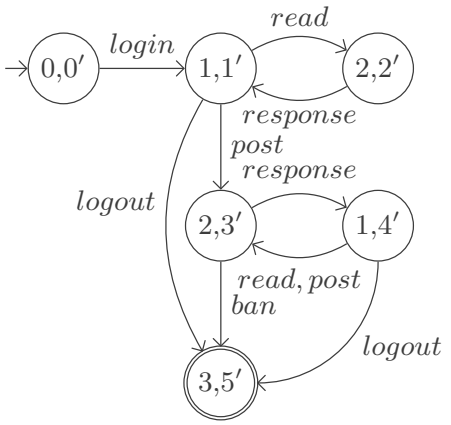
For example, consider the MAS *ROServer* in Fig. 6. This specification corresponds to a read-only forum: logged users may read messages and log out, but if they try to post a message, the server will ban them.




$\mathrm{Acc}(0') = \{\{login\}\}$
$\mathrm{Acc}(1') = \{\{read, post\}, \{read, post, logout\}\}$
$\mathrm{Acc}(2') = \{\{response\}\}$
$\mathrm{Acc}(3') = \{\{response, ban\}\}$
$\mathrm{Acc}(4') = \{\{read, post\}, \{read, post, logout\}\}$
$\mathrm{Acc}(5') = \{\emptyset\}$

**Fig. 6.** Specification *ROServer*

Now the Fig. 7 shows the quotient of *Server* and *ROServer*. Note that a priority $P$ is generated to enforce the eventual choice of the transition labeled *ban* in any model $M_{/}$ and thus guarantee that, regardless of the implementation choices that are made when building a model $M$ of *ROserver*, $M \times M_{/}$ will satisfy the reachability constraint included in *Server*. A screenshot of MAccS with the result of this quotient is shown in Fig. 8.




$$\begin{aligned}
&\mathrm{Acc}(0,0') = \{\{login\}\}\\
&\mathrm{Acc}(1,1') = \{\{read, post\}, \{read, post, logout\}\}\\
&\mathrm{Acc}(2,2') = \{\{response\}\}\\
&\mathrm{Acc}(2,3') = \{\{response\}, \{response, ban\}\}\\
&\mathrm{Acc}(1,4') = \{\{read, post\}, \{read, post, logout\}\}\\
&\mathrm{Acc}(3,5') = \{\emptyset\}
\end{aligned}$$

$$P = \{\{((2,3'), ban)\}\}$$

**Fig. 7.** Result of *Server*/*ROServer*

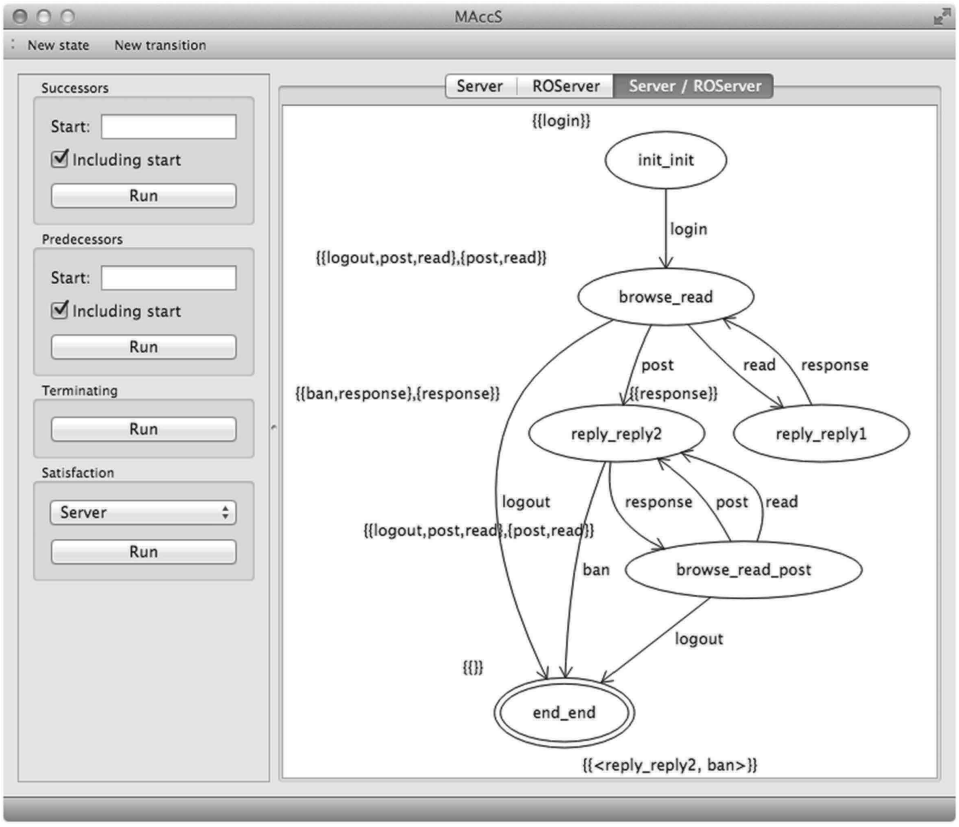



**Fig. 8.** Screenshot of MAccS with the result of *Server*/*ROServer*

*Conjunction and consistency of MAS.* It is a current practice to attach to a given (sub)system several specifications, each of them describing a different aspect or viewpoint of the (sub)system. These specifications have to be interpreted in a conjunctive way.

The tool MAccS addresses viewpoint-design by implementing a conjunction operator for MAS: given two MAS $S_1$ and $S_2$, the MAS $S_1 \wedge S_2$, whose set of models is exactly the intersection of the set of models of $S_1$ and $S_2$, can be computed. If this intersection is empty, the two MAS are declared inconsistent.

## 3 Related Work

MAS extend modal transition systems (MTS) [2] by allowing to specify any combination of actions and not only the sets of actions which belong to an interval defined by may/must transitions. Moreover, the possibility to require some reachability constraints on states is only available for MAS and also improves expressivity with respect to MTS.

MAccS is the first tool for MAS with the support of a complete specification theory. MTS have been identified as a specification formalism particularly suitable for interface-based design [3,6], contract-based design [7], software product lines description [8] and model merging [9] but only a few tools support them. The tool MTSA [10] supports refinement, product and conjunction of MTS but no quotient. The tools MIO workbench [11] and MICA [12] are dedicated to MTS enriched with input and output actions. They both support a complete specification theory but restricted to the modal case and their operators do not guarantee concurrent reachability. The tool MoTRAS [13] also proposes a complete specification theory for MTS, which are less expressive than MAS. MoTRAS allows for LTL model-checking of specifications but this is not enough to go along with reachability constraints as advocated in [1]. Several non-deterministic MTS variants, namely Disjunctive MTS [14], Boolean MTS and Parametric MTS [15], which have a similar expressive power as acceptance specifications (without marked states), are also under the scope of MoTRAS, but only for a reduced set of operations: there is no quotient for DMTS and only refinement and the deterministic hull for BMTS and PMTS. Last, the tools ECDAR [16] and PyECDAR [17] support a complete timed specification theory.

## References


1. Verdier, G., Raclet, J.-B.: Quotient of acceptance specifications under reachability constraints. In: LATA 2015. LNCS, vol. 8977 (2015, to appear)
2. Larsen, K.G., Thomsen, B.: A modal process logic. In: LICS, pp. 203–210, IEEE (1988)
3. Raclet, J.B., Badouel, E., Benveniste, A., Caillaud, B., Legay, A., Passerone, R.: A modal interface theory for component-based design. Fundam. In. **108**(1–2), 119–149 (2011)
4. Siek, J., Lee, L.Q., Lumsdaine, A.: The Boost Graph Library. Addison-Wesley, Boston (2002)

5. Gansner, E.R., North, S.C.: An open graph visualization system and its applications to software engineering. Softw. Pract. Exp. **30**(11), 1203–1233 (2000)
6. Larsen, K.G., Nyman, U., Wąsowski, A.: Modal I/O automata for interface and product line theories. In: De Nicola, R. (ed.) ESOP 2007. LNCS, vol. 4421, pp. 64–79. Springer, Heidelberg (2007)
7. Benveniste, A., Caillaud, B., Nickovic, D., Passerone, R., Raclet, J.-B., Reinkemeier, P., Sangiovanni-Vincentelli, A., Damm, W., Henzinger, T., Larsen, K.G.: Contracts for system design. Research report, RR-8147, 65 pp., Nov 2012. https://hal.inria.fr/hal-00757488
8. Asirelli, P., ter Beek, M.H., Fantechi, A., Gnesi, S.: A logical framework to deal with variability. In: Méry, D., Merz, S. (eds.) IFM 2010. LNCS, vol. 6396, pp. 43–58. Springer, Heidelberg (2010)
9. Uchitel, S., Chechik, M.: Merging partial behavioural models. In: SIGSOFT FSE, pp. 43–52, ACM (2004)
10. D'Ippolito, N., Fischbein, D., Chechik, M., Uchitel, S.: MTSA: The modal transition system analyser. In: ASE, pp.475–476, IEEE (2008)
11. Bauer, S.S., Mayer, P., Legay, A.: MIO workbench: a tool for compositional design with modal input/output interfaces. In: Bultan, T., Hsiung, P.-A. (eds.) ATVA 2011. LNCS, vol. 6996, pp. 418–421. Springer, Heidelberg (2011)
12. Caillaud, B.: Mica: a modal interface compositional analysis library (Oct 2011). http://www.irisa.fr/s4/tools/mica
13. Křetínský, J., Sickert, S.: MoTraS: a tool for modal transition systems and their extensions. In: Van Hung, D., Ogawa, M. (eds.) ATVA 2013. LNCS, vol. 8172, pp. 487–491. Springer, Heidelberg (2013)
14. Larsen, K.G., Xinxin, L.: Equation solving using modal transition systems. In: Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science, pp. 108–117, Philadelphia, Pennsylvania, USA, 4–7 June 1990
15. Beneš, N., Křetínský, J., Larsen, K.G., Møller, M.H., Srba, J.: Parametric modal transition systems. In: Bultan, T., Hsiung, P.-A. (eds.) ATVA 2011. LNCS, vol. 6996, pp. 275–289. Springer, Heidelberg (2011)
16. David, A., Larsen, K.G., Legay, A., Nyman, U., Wąsowski, A.: ECDAR: an environment for compositional design and analysis of real time systems. In: Bouajjani, A., Chin, W.-N. (eds.) ATVA 2010. LNCS, vol. 6252, pp. 365–370. Springer, Heidelberg (2010)
17. Legay, Axel, Traonouez, Louis-Marie: PyEcdar: towards open source implementation for timed systems. In: Van Hung, Dang, Ogawa, Mizuhito (eds.) ATVA 2013. LNCS, vol. 8172, pp. 460–463. Springer, Heidelberg (2013)