EB[ASTD]: Meta-modelling framework for ASTD

Christophe Chen[1,2], Peter Rivière[1,3], Neeraj Kumar Singh[1],
Guillaume Dupont[1], Yamine Ait Ameur[1], Marc Frappier[2]

[1] *INPT-ENSEEIHT/IRIT, University of Toulouse, France*
[2] *Université de Sherbrooke, Sherbrooke, QC, Canada*
[3] *JAIST - Japan Advanced Institute of Science and Technology, Ishikawa, Japan*
{christophe.chen, peter.riviere, nsingh, guillaume.dupont,
yamine}@enseeiht.fr
marc.frappier@usherbrooke.ca

Algebraic State Transition Diagram (ASTD) [8] is a formal, graphical, state-based modeling language designed for the development of complex critical systems [10,1]. It provides a set of process algebra operators to compose hierarchical state machines, streamlining modularity in system design. Furthermore, its operational semantics defines transition rules for each ASTD operator.

Despite advances in incorporating features such as local state variables [13] and real-time [2], ASTD tool support (cASTD [13,14], pASTD [4], ASTD2EB [7,9]) is based on *ad hoc* model transformation that do not preserve the original structure of ASTDs. Each technology offers custom tools for ASTD, which can introduce new errors and complicate the integration of multiple tools on the same model. Moreover, it is essential to identify conditions characterizing well-defined ASTD, and to establish new properties (e.g., liveness) taking into account their operational semantics. In fact, our main objective is to seek a generic approach for *formally* **reasoning** on ASTD models.

The meta-model. In this context, we introduce EB[ASTD], an algebraic meta-model of ASTD formalizing its operational semantics [6]. This ground model, inspired from the EB4EB methodology [11,12], relies on a deep modelling strategy that integrates the ASTD syntax and semantics as Event-B algebraic theories, as described below. The whole model can be found at `https://www.irit.fr/EBRP/software/`

a) *Syntax and static semantics.* Listing 1 presents the `ASTDStruct` Event-B theory, which describes ASTD in a denotational style. Each compositional operator of ASTD is included in the datatype as a constructor, while component access is handled through destructors. Several predicate operators define the static semantics for ASTD to verify well-instantiation. These operators can generate proof obligations (POs) automatically by setting them as theorem in the Event-B context.

b) *Operational semantics.* Listing 2 defines the operational semantics of ASTD.

```
THEORY ASTDStruct
TYPE PARAMETERS St, Ev, Var
DATA TYPES
ASTD(St, Ev, Var)
 constructors
   Elementary(...)
   Automaton(...)
   Sequence(...)
   Closure(...)
   Guard(...)
OPERATORS
Invariant_WellCons predicate
Scope_WellCons predicate
...
ASTD_WellCons predicate
(a : ASTD(St, Ev, Var), accVar : ℙ(Var))
```

Listing 1. ASTD syntax

Several transitions rules are defined for each ASTD operator, and these rules are incorporated in the operator *NextState*. Given an ASTD *astd*, its current state *curr* and a triggered event σ, a set of all possible next states is returned according to the operational semantics. For example, in the case of Automaton, three enumerated set aut_i defined to represent the transition rules for aut_i.

```
NextState expression (
    astd : ASTD(St, Ev, Var) , σ : Ev ,
    curr : ASTDState(St, Var))
well−definedness condition ...
recursive definition
case astd :
    Elementary(inv) => ...
    Automaton(i, f, ..., inv, mapping) =>
        aut1 ∪ aut2 ∪ aut3
    Sequence(fst, snd, attr, initAttr, inv)
        => ....
    Closure(...) => ...
    Guard(...) => ...
```

Listing 2. ASTD Transitions rules

c) *Two instantiation mechanisms.* We use both the deep and shallow instantiation mechanisms. The deep approach models ASTD as an instance of the meta-model, encoding it as a first-class object. In contrast, the shallow approach leverages the operational semantics of ASTD to generate the initial state and manage state changes. This enables using the ProB model checker and the visual animator VisB for validating ASTDs.

Proof-based reasoning. The framework enables the definition of proof obligations, checking their soundness and generating them. We illustrate this approach using POs for state invariants defined in pASTD [5]. Their POs lacks a formal justification, i.e. that they adequately represent their associated property. To achieve this, we encode the specification of POs in the form of properties on traces, allowing us to demonstrate that $ASTD \vdash POs \implies Spec[PO]\_On\_Traces$. For instance, invariant preservation corresponds to $ASTD \vdash PO_{pASTD} \implies \forall tr \in Traces(ASTD), INV(tr(i))$, i.e., that every execution of the ASTD satisfies the invariant when PO_{pASTD} hold. Three theories are defined for this.

a) *Proof obligations.* The definition and the generation process is straightforward. It requires the direct definition of the PO within the $ASTDPO$ theory operator. Then the operator is used as theorem in Event-B context, entailing PO generation. b) *Trace-based semantics.* The definition of traces exploits the operational semantics of ASTD in the theory $ASTDTraces$.

$$
\begin{array}{l}
thm\_of\_PO\_Correctness: \\
\forall astd, tr \cdot \\
astd \in ASTD(St, Ev, Var) \\
\land ASTD\_WellCons(astd, \emptyset) \\
\land tr \in \mathbb{N} \pfun ASTDState(St, Var) \\
\land IsATrace(astd, tr) \\
\land PO_{pASTD}(astd, ...) \\
\Rightarrow \\
(\forall i \cdot i \in dom(tr) \Rightarrow INV(astd, tr(i)))
\end{array}
$$

Listing 3. Proof of soundness

c) *Soundness.* The last theory $ASTDCorrectness$ leverages the specification of invariant satisfaction for a given state ($INV(tr(i))$. Finally, the soundness theorem is defined and proved in Listing 3. This proof highlighted some bugs in the manual specification [5,3]. Interested reader can consult [6] for more details.

Conclusion. This framework offers an explicit manipulation of ASTD concepts as first-class citizens. It features a proof-based mechanism that enables reasoning on specific ASTDs defined as instances of this meta-model. The tool is built upon the Rodin platform, which provides automated proof obligation generation, automatic and interactive verification, graphical animation, and model checking of ASTDs. Overall, the EB[ASTD] framework provides a sound foundation for proving properties about ASTDs and operates effectively within the Rodin platform, which is specifically designed for managing ASTD models and proofs.

References

1. de Azevedo Oliveira, D., Frappier, M.: Modelling an automotive software system with TASTD. In: Glässer, U., Campos, J.C., Méry, D., Palanque, P.A. (eds.) Rigorous State-Based Methods - 9th International Conference, ABZ 2023, Nancy, France, May 30 - June 2, 2023, Proceedings. Lecture Notes in Computer Science, vol. 14010, pp. 124–141. Springer (2023), https://doi.org/10.1007/978-3-031-33163-3_10
2. de Azevedo Oliveira, D., Frappier, M.: TASTD: A real-time extension for ASTD. In: Glässer, U., Campos, J.C., Méry, D., Palanque, P.A. (eds.) Rigorous State-Based Methods - 9th International Conference, ABZ 2023, Nancy, France, May 30 - June 2, 2023, Proceedings. Lecture Notes in Computer Science, vol. 14010, pp. 142–159. Springer (2023), https://doi.org/10.1007/978-3-031-33163-3_11
3. de Azevedo Oliveira, D., Frappier, M.: TASTD: A real-time extension for ASTD. In: Glässer, U., Campos, J.C., Méry, D., Palanque, P.A. (eds.) Rigorous State-Based Methods - 9th International Conference, ABZ 2023, Nancy, France, May 30 - June 2, 2023, Proceedings. Lecture Notes in Computer Science, vol. 14010, pp. 142–159. Springer (2023), https://doi.org/10.1007/978-3-031-33163-3_11
4. Cartellier, Q., Frappier, M., Mammar, A.: Proving local invariants in ASTDs. In: Li, Y., Tahar, S. (eds.) Formal Methods and Software Engineering - 24th International Conference on Formal Engineering Methods, ICFEM 2023, Brisbane, QLD, Australia, November 21-24, 2023, Proceedings. Lecture Notes in Computer Science, vol. 14308, pp. 228–246. Springer (2023), https://doi.org/10.1007/978-981-99-7584-6_14
5. Cartellier, Q., Frappier, M., Mammar, A.: Proving local invariants in ASTDs. In: Li, Y., Tahar, S. (eds.) Formal Methods and Software Engineering - 24th International Conference on Formal Engineering Methods, ICFEM 2023, Brisbane, QLD, Australia, November 21-24, 2023, Proceedings. Lecture Notes in Computer Science, vol. 14308, pp. 228–246. Springer (2023), https://doi.org/10.1007/978-981-99-7584-6_14
6. Chen, C., Rivière, P., Singh, N.K., Dupont, G., Ait Ameur, Y., Frappier, M.: A proof-based ground algebraic meta-model for reasoning on ASTD in Event-B. In: 13th International Conference on Formal Methods in Software Engineering (FormaliSE) (2025)
7. Fayolle, T., Frappier, M., Laleau, R., Gervais, F.: Formal refinement of extended state machines. In: Derrick, J., Boiten, E.A., Reeves, S. (eds.) Proceedings 17th International Workshop on Refinement, Refine@FM 2015, Oslo, Norway, 22nd June 2015. EPTCS, vol. 209, pp. 1–16 (2015), https://doi.org/10.4204/EPTCS.209.1
8. Frappier, M., Gervais, F., Laleau, R., Milhau, J.: Refinement patterns for ASTDs. Formal Aspects Comput. 26(5), 919–941 (2014), https://doi.org/10.1007/s00165-013-0286-3
9. Milhau, J., Frappier, M., Gervais, F., Laleau, R.: Systematic translation rules from ASTD to Event-B. In: Méry, D., Merz, S. (eds.) Integrated Formal Methods - 8th International Conference, IFM 2010, Nancy, France, October 11-14, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6396, pp. 245–259. Springer (2010), https://doi.org/10.1007/978-3-642-16265-7_18
10. Ndouna, A.R., Frappier, M.: Modelling a mechanical lung ventilation system using TASTD. In: Bonfanti, S., Gargantini, A., Leuschel, M., Riccobene, E., Scandurra, P. (eds.) Rigorous State-Based Methods - 10th International Conference, ABZ 2024, Bergamo, Italy, June 25-28, 2024, Proceedings. Lecture Notes in Computer Science, vol. 14759, pp. 324–340. Springer (2024), https://doi.org/10.1007/978-3-031-63790-2_26

11. Rivière, P., Singh, N.K., Aït-Ameur, Y.: Reflexive Event-B: Semantics and correctness the EB4EB framework. IEEE Trans. Reliab. 73(2), 835–850 (2024), https://doi.org/10.1109/TR.2022.3219649
12. Riviere, P., Singh, N.K., Aït-Ameur, Y., Dupont, G.: Formalising Liveness Properties in Event-B with the Reflexive EB4EB Framework (2023)
13. Tidjon, L.N., Frappier, M., Leuschel, M., Mammar, A.: Extended algebraic state-transition diagrams. In: 23rd International Conference on Engineering of Complex Computer Systems, ICECCS 2018, Melbourne, Australia, December 12-14, 2018. pp. 146–155. IEEE Computer Society (2018), https://doi.org/10.1109/ICECCS2018.2018.00023
14. Tidjon, L.N., Frappier, M., Mammar, A.: Intrusion detection using ASTDs. In: Barolli, L., Amato, F., Moscato, F., Enokido, T., Takizawa, M. (eds.) Advanced Information Networking and Applications - Proceedings of the 34th International Conference on Advanced Information Networking and Applications, AINA-2020, Caserta, Italy, 15-17 April. Advances in Intelligent Systems and Computing, vol. 1151, pp. 1397–1411. Springer (2020), https://doi.org/10.1007/978-3-030-44041-1_118