Formalization in Matita of the Paper Causal Reversibility Implies Time Reversibility (A QEST2023 artifact submission)

Marco Bernardo[1], Ivan Lanese[2], Andrea Marin[3],
Claudio A. Mezzina[1], Sabina Rossi[3], and Claudio Sacerdoti Coen[4]

[1] Department of Pure and Applied Sciences, University of Urbino, Urbino, Italy
[2] Focus Team, University of Bologna & INRIA, Bologna, Italy
[3] Dept. of Env. Sciences, Informatics and Statistics, Univ. Ca' Foscari, Venice, Italy
[4] Dept. of Informatics – Science and Engineering, Univ. of Bologna, Bologna, Italy

Abstract. We present a formalization in the Interactive Theorem Prover Matita of the main result of the paper, i.e. Theorem 2. This theorem states that if a Reversible Markovian Labeled Transition System with Independence (RMLTSI) meets Causal Consistency (CC) and Product Preservation along Squares (PPS), then for all cycles ω we have that the product of rates over ω equals the product of rates over ω traversed in opposite direction. The formalization also includes all the definitions and lemmas required to state and prove Theorem 2, including an axiomatization and a model of the additive group of integer numbers, which is missing from the standard library of Matita.

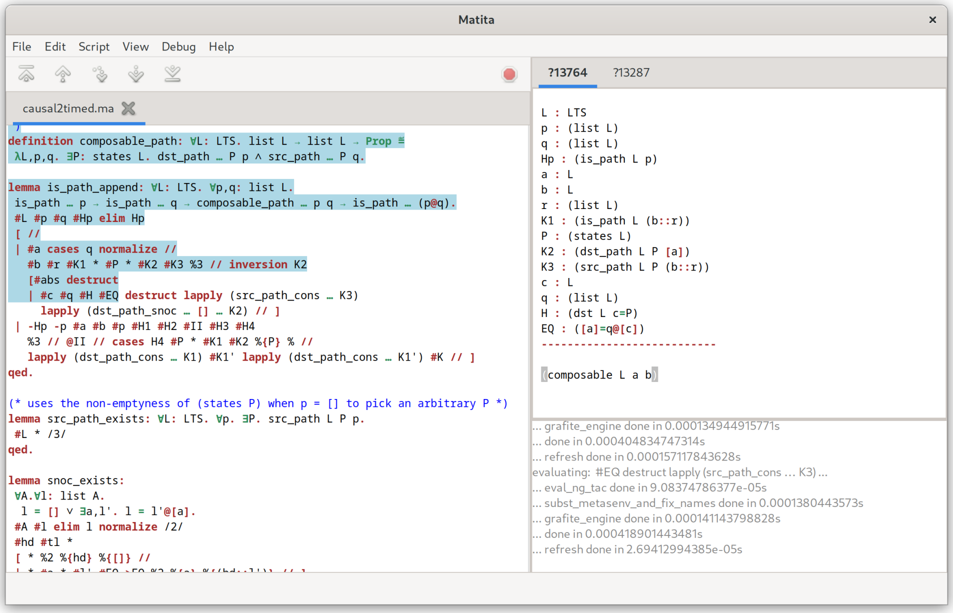
1 Matita

Matita [1] is an Interactive Theorem Prover (ITP) developed at the University of Bologna and based on a variant of the Calculus of (Co)Inductive Constructions (see for example the nice exposition in Chapter 3 of [2] for one of the variants). Therefore it behaves similarly to other ITPs based on similar type theories, like Coq, Lean and Agda.

It allows to give definitions and prove theorems about them, checking that all definitions are well-formed — in particular that all functions are total — and that all proofs hold.

In Figure 1 we show a screenshot of the user interface of Matita. The text of the formalization is in the left pane. Blue lines have already been checked, while the white lines below are not checked yet. The last two buttons below the menu bar, which show arrows pointing down, ask the system to check the next command or the remaining of the file. Since checked (blue) lines are immutable, the first two buttons ask the system to forget that some part of the text has already been checked in order to modify it.

When checking is stopped in the middle of a proof, like in the screenshot, the right pane shows the set of remaining proof obligations, made of a list of hypotheses above the dashed horizontal line and the wanted conclusion below



```
)
definition composable_path: ∀L: LTS. list L → list L → Prop ≝
 λL,p,q. ∃P: states L. dst_path … P p ∧ src_path … P q.

lemma is_path_append: ∀L: LTS. ∀p,q: list L.
 is_path … p → is_path … q → composable_path … p q → is_path … (p@q).
 #L #p #q #Hp elim Hp
 [ //
 | #a cases q normalize //
   #b #r #K1 * #P * #K2 #K3 %3 // inversion K2
   [#abs destruct
   | #c #q #H #EQ destruct lapply (src_path_cons … K3)
     lapply (dst_path_snoc … [] … K2) // ]
 | -Hp -p #a #b #p #H1 #H2 #II #H3 #H4
   %3 // @II // cases H4 #P * #K1 #K2 %{P} % //
   lapply (dst_path_cons … K1) #K1' lapply (dst_path_cons … K1') #K // ]
qed.

(* uses the non-emptyness of (states P) when p = [] to pick an arbitrary P *)
lemma src_path_exists: ∀L: LTS. ∀p. ∃P. src_path L P p.
 #L * /3/
qed.

lemma snoc_exists:
 ∀A.∀l: list A.
  l = [] ∨ ∃a,l'. l = l'@[a].
 #A #l elim l normalize /2/
 #hd #tl *
 [ * %2 %{hd} %{[]} //
 | * #a * #l' #EQ >EQ %2 %{a} %{(hd::l')} // ]
```

```
L : LTS
p : (list L)
q : (list L)
Hp : (is_path L p)
a : L
b : L
r : (list L)
K1 : (is_path L (b::r))
P : (states L)
K2 : (dst_path L P [a])
K3 : (src_path L P (b::r))
c : L
q : (list L)
H : (dst L c=P)
EQ : ([a]=q@[c])
__________________________

(composable L a b)
```

```
... grafite_engine done in 0.000134944915771s
... done in 0.000404834747314s
... refresh done in 0.000157117843628s
evaluating: #EQ destruct lapply (src_path_cons … K3) ...
... eval_ng_tac done in 9.08374786377e-05s
... subst_metasenv_and_fix_names done in 0.0001380443573s
... grafite_engine done in 0.000141143798828s
... done in 0.000418901443481s
... refresh done in 2.69412994385e-05s
```

Fig. 1: The user interface of Matita

the line. Each proof obligation is shown in its own tab and there are two in the screenshot.

The proof style is procedural: while definitions are directly human readable, proof steps are just commands, called tactics, to instruct the system to advance in the proof and therefore are not meant to be understood by a user without replaying the proof one step at a time. This is however not necessary to trust the proof: once the statement and definitions have been verified and Matita has checked the whole file, the results in them hold (up to bugs in the implementation of Matita, of course).

2 The Formalization

We formalized all definitions and lemmas required to verify Theorem 2 of the paper, which states that if a Reversible Markovian Labeled Transition System with Independence (RMLTSI) meets Causal Consistency (CC) and Product Preservation along Squares (PPS), then for all cycles ω we have that the product of rates over ω equals the product of rates over ω traversed in opposite direction. The final result of the paper, Corollary 1, follows from Theorem 2 by simply applying a well-known result from the literature.

Paper	Formalization
Definition 1 (reversible LTS with independence)	`RLTSI`
Definition 2 (commuting square)	`commuting_square`
Definition 3 (propagation of coinitial independence)	`PCI` (in complements.ma)
Definition 4 (causal equivalence)	`equiv`
Proposition 1	`equiv_count_action`
Definition 5 (causal consistency)	`CC`
Definition 6 (uniqueness of pairs) + Proposition 2	`uniqueness_of_pairs`
Definition 7 (absence of self-loops) + Proposition 3	`absence_of_self_loops`
Definition 10 (reversible Markovian LTS with independence)	`RMLTSI`
rateprod	`rateprod`
Definition 11 (product preservation along squares)	`PPS`
Lemma 1	`rateprod_rev`
Theorem 2	`main`

Table 1: Correspondence between the paper and the formalization

The formalization consists of several files:

- causal2timed.ma: it contains all the definitions and proofs in the paper required for Theorem 2, which is the main result of the paper that is formalized in Matita
- z_axioms.ma: the previous file depends on this one that provides an axiomatization of integer numbers
- z.ma: this file shows a model of the axioms in z_axioms.ma (or, equivalently, an implementation of integer numbers that satisfies the axiomatization given in z_axioms and used in the proof). It is not required by causal2timed.ma and therefore it needs to be checked independently.
- complements.ma: this file contains the proof that one of the hypotheses in the paper, propagation of coinitial independence, is redundant, but could be used to simplify another hypothesis. It is not required by causal2timed.ma and therefore it needs to be checked independently.

Table 1 shows the correspondence between all definitions and theorems of Section 2.1 (Causal Reversibility of Concurrent Systems) and Section 3 (Relationships between Causal and Time Reversibilities) of the paper and the corresponding ones in the formalization. Section 2.2 (Time Reversibility of Continuous-Time Markov Chains) has not been formalized because it contains only standard definitions and because it is used in the paper only to derive Corollary 1, which is a trivial consequence of Theorem 2 — that is formalized — and Theorem 1, which is standard.

2.1 Definitions in the Formalization

We list now all the definitions in the formalization, explaining how they relate to the ones in the paper. From now on we assume that the evaluator of the artifact is acquainted with ITPs and, in particular, with those based on variants of the Calculus of Construction or Martin-Löf type theory.

`LTS:` An LTS (Labeled Transition System) is the mathematical structure that formalizes the notion of Labeled Transition System representing states and actions using two distinct types `states` and `actions`. With a minor loss of generality w.r.t. the paper[5], we assume a decision procedure `eql` for the equality of actions. The type of states is forced to be non-empty by asking for a distinguished process `a_state` that plays no particular role.

In the paper transitions are triples; they are formalized with another type `transitions` equipped with three projections `src/dst/action_of` to retrieve the components of the triple, together with the assumption that transitions having the same projections are equal.

`LTSI` An LTSI (a Labeled Transition System with Independence) is a LTS that additionally introduces a symmetric independence relation over transitions. The irreflexivity hypothesis in the paper is dropped because not necessary for Theorem 2.

`RLTSI` An RLTSI (Reversible LTS with Independence) captures Definition 1 in the paper. It adds to a LTSI a boolean function `is_fw` to assign a direction to transitions. We also add a `rev` function to return the reverse of a transition, capturing the Loop Property in the paper, and we assume some basic properties over `rev`, like the swapping of the source/destination of the transition or the flipping of the direction.

There is a minor difference between RLTSIs in the paper and in the formalization. The paper considers the (disjoint) union of two transition relations, one going forward and one going backward, and it assumes to be able to distinguish between elements of them when needed. This is what is tipically done in all the papers that consider reversible LTS, and we decided to stick to it. However, it introduces an abuse that needs to be rectified in formalizations. Indeed, the transition relation of an LTS is made of triples (s, a, s'). Therefore one cannot just formalize elements of the disjoint union as pairs $(b, (s, a, s'))$ where $b \in \{true, false\}$ records if the transition is going forward or backward: the result would not be an LTS anymore. However, one can note that elements of the form $(b, (s, a, s'))$ are isomorphic to elements of the form $(s, (b, a), s')$ and obtain again an LTS where the direction of the transition is encoded in the action. This is what we did in the formalization. Concretely, the only differences w.r.t. the paper are the following:

1. we add to RLTSI a new hypothesis `disjoint_fw_bw` claiming that transitions going in opposite directions have distinct actions: this is a way to implicitly distinguish between forward actions $(true, a)$ and backward actions $(false, a)$ abstracting over the internal structure of an action (actions remains an abstract type and not pairs whose first element is a boolean)
2. instead of assuming that a transition and its reverse have the same action as it is written in the paper, we add to RLTSI a new hypothesis `action_rev`

[5] The logic of Matita is intuitionistic. Therefore the formalization shows that the proof in the paper is constructive. In classical logic all types have decidable equalities.

that just states that two transitions have the same action iff their reverse transitions also have the same action. It turns out that this is a sufficient condition for all proofs in the paper

coinit/cofinal/composable are predicates relating two transitions, defined as in the paper.

is_path: the paper assumes paths to be always made of consecutive transitions. In the formalization we represent paths using lists of transitions and we use the predicate `is_path` to identify those lists that are made of consecutive transitions only. Note that some theorems in the formalization hold over arbitrary lists of transitions, without assuming `is_path`.

Something must be said on the representation of empty paths. An empty path is usually thought of as *pointed*, i.e. originating and terminating in the same state, and two empty paths pointed on different states are different. In the formalization, to avoid introducing a special case for empty paths, we just represent an empty path with an empty list of transitions. Therefore our empty paths are not pointed and comparing two empty paths is not to be done. Interestingly, our proofs never need to compare paths.

dst_path/src_path/coinit_path/cofinal_path/composable_path : generalize the relations with the same names from transitions to paths. The only tricky point is that, since empty paths are not pointed, an empty path originates and terminates on any state — the `dst_path/src_path` relations are one-to-many on empty paths — and therefore an empty path is coinitial/cofinal/consecutive to any other path. It also happens that an empty path can at once originate in a state and terminate in another. This contradicts pointedness, but it is actually totally innocuous because it cannot be used to form paths that do not satisfy `is_path`: concatenating a non-empty path (i.e. a list that satisfies `is_path`) `p` with an empty path yields again `p` that terminates (only) in the final state of `p` even if the empty path was terminating on any state.

is_cycle/commuting_square : follow the corresponding definitions in the paper. Note that cycles are required to be non-empty paths, i.e. they are non-empty and they also satisfy `is_path`.

rev_path computes the reverse of a list of transitions. Lemma `rev_path_wf` grants that the reversal of a path, i.e. a `is_path` list of transitions, is still a path.

equiv/equiv' are alternative formalizations of the notion of causal equivalence and they are proved to be equivalent.

match_action/count_action : `match_action` takes an action and a transition and returns +1 if the action is the one of the transition, -1 if it is the action of the reverse of the transition, 0 otherwise. It is used to define `count_action` that extends the definition from a transition to a path, returning the difference between the number of occurrences of an action in the path and the number of occurrences of the corresponding reverse action. This quantity is proved to be equal for causal equivalent paths.

CC follows the definition of causal consistency in the paper.

specific_cancellative_monoid : it is the definition of an algebraic structure that generalizes cancellative monoids. In particular any specific cancellative monoid is a cancellative monoid and any cancellative monoid has a specific cancellative submonoid. As far as we know, there is no standard name for this algebraic structure in the literature. The structure captures the minimal set of operations over rates that is required for Theorem 2 to hold. Every commutative cancellative monoid is a specific cancellative monoid and therefore positive real numbers, used for rates in Markov chains, are a specific cancellative monoid.

RMLTSI : an RMLTSI (Reversible Markovian LTS with Independence) is a `RLTSI` with an additional function `rate` that maps transitions to rates (elements of a specific cancellative monoid). It captures Definition 10 in the paper.

PPS follows Definition 11 in the paper.

2.2 The Main Theorem

Theorem `main` in the formalization corresponds to Theorem 2 in the paper. The two statements are identical, up to the additional hypothesis `is_path $\ldots$ p` in the formalization to assume the list of transitions `p` to be a path.

2.3 On the Formalization of Integer Numbers

Integer numbers, used in the proof to count the number of actions with a certain label and direction in a path, are not part of the standard library of Matita. The file z_axioms.ma provides an axiomatization based on zero, successor, predecessor, addition and their properties. An axiom-free model that does not require quotients is given in file z.ma. The main proof is based on the axiomatization and not on the particular model.

3 Conclusions

The main theorem of the paper holds. Moreover, it works on a slight generalization of the hypotheses in the paper: the independence relation is not required to be irreflexive, the set of actions is not required to be non-empty and the propagation of coinitial independence (PCI) hypothesis is not required as well.

The proof in the file complements.ma shows that assuming PCI one could simplify Definition 11 (product preservation along squares, `PPS` in the formalization) by only requiring the half of the definition that deals with forward transitions. Indeed the requirement on backward transitions would be a consequence of the reverse of a commuting square being again a commuting square (theorem `reverse_square` in file complements.ma).

Acknowledgments. This work has been supported by the Italian PRIN 2020 project *NiRvAna – Noninterference and Reversibility Analysis in Private Blockchains*, the French ANR-18-CE25-0007 project *DCore – Causal Debugging for Concurrent Systems*, and the INdAM-GNCS E53C22001930001 project *RISICO – Reversibilità in Sistemi Concorrenti: Analisi Quantitative e Funzionali*.

References

1. Asperti, A., Ricciotti, W., Sacerdoti Coen, C.: Matita tutorial. Journal of Formalized Reasoning **7**, 91–199 (2014)
2. Lennon-Bertrand, M.: Bidirectional Typing for the Calculus of Inductive Constructions. (Typage Bidirectionnel pour le Calcul des Constructions Inductives). Ph.D. thesis, University of Nantes, France (2022), `https://tel.archives-ouvertes.fr/tel-03848595`