On (de-)composing causality*

Georgiana Caltais[1], Stefan Leue[1], and Mohammad Reza Mousavi[2]

[1] Department for Computer and Information Science, University of Konstanz, Germany
[2] Centre for Research on Embedded Systems, Halmstad University, Sweden

Abstract

This work introduces a notion of counterfactual causality in the Halpern and Pearl sense that is compositional with respect to the interleaving of non-communicating transition systems. The underlying logic is Hennessy Milner logic. This is an abstract based on a paper accepted in CREST 2016.

Introduction. Determining and computing causalities is a frequently addressed issue in the philosophy of science and engineering, for instance when causally relating system faults to system failures. A notion of causality that is frequently used in relation to technical systems relies on counterfactual reasoning [17]. In short, the counterfactual argument defines when an event is considered a cause for some effect, in the following way: a) whenever the event presumed to be a cause occurs, the effect occurs as well, and b) when the presumed cause does not occur, the effect will not occur either. The seminal paper [11] describes an event model and a notion of actual causation encompassing the counterfactual argument. Most relevant for our work are the contributions in [16, 15]. The latter provide an interpretation of the results in [11] in the context of transition systems and trace models for concurrent system computations.

The objective of this paper is to consider the notion of counterfactual causality reasoning and actual causation in the context of labeled transition systems (LTS's). In our setting the LTS's represent system models and Hennessy Milner logic (HML) [12] formulae specify the system properties for whose violation actual causes are sought. We also establish first results on computing causalities in this setting using (de-)compositional verification. This is an abstract of the paper [3], accepted in CREST 2016[1]. For the complete definitions and proofs we refer the interested reader to [3].

Preliminaries. Next, we provide a brief overview of LTS's and their computations, and HML. A *labeled transition system* (LTS) is a triple $T = (\mathbb{S}, s_0, A, \to)$, where $\mathbb{S}$ is the set of states, $s_0 \in \mathbb{S}$ is the initial state, A is the action alphabet and $\to \subseteq \mathbb{S} \times A \times \mathbb{S}$ is the transition relation. We write $\twoheadrightarrow \subseteq \mathbb{S} \times A^* \times \mathbb{S}$, to denote the reflexive and transitive closure of $\to$.

Let $\mathcal{D}$, $\mathcal{D}_i$ range over possibly infinite lists of words in A^*; we write ε for the empty word. We say that such lists are *size-compatible* if they are finite lists of the same length, or if they are all infinite lists. Let $\pi = (s_0, l_0, \mathcal{D}_0), \dots (s_n, l_n, \mathcal{D}_n), s_{n+1} \in (\mathbb{S} \times A \times [A^*])^* \times \mathbb{S}$. Assume that $\mathcal{D}_0, \dots, \mathcal{D}_n$ are size-compatible. Intuitively, we write $traces(\pi)$ to denote the pairwise extensions of $l_0 \dots l_n$ with words "at the same level" in $\mathcal{D}_0, \dots, \mathcal{D}_n$. For instance, if $\pi = (s_0, l_0, [w_1^0, w_2^0]), (s_1, l_1, [w_1^1, w_2^1]), s_2$, then $traces(\pi) = \{l_0 w_1^0 l_1 w_1^1,\ l_0 w_2^0 l_1 w_2^1\}$. We say that π is a *computation* of T whenever the following hold: (i) $s_0 \xrightarrow{l_0} s_1 \dots \xrightarrow{l_n} s_{n+1}$, (ii) $\mathcal{D}_0, \dots, \mathcal{D}_n$ are size-compatible, and (iii) for all $w \in traces(\pi)$ there exists $s \in \mathbb{S}$ such that $s_0 \overset{w}{\twoheadrightarrow} s$. $sub(\pi)$

*The work of Georgiana Caltais was partially supported by an Independent Research Start-up Grant founded by Zukunftskolleg at Konstanz University. The work of Mohammad Reza Mousavi has been partially supported by the Swedish Research Council (Vetenskapsrådet) award number: 621-2014-5057 (Effective Model-Based Testing of Concurrent Systems) and the Swedish Knowledge Foundation (Stiftelsen för Kunskaps- och Kompetensutveckling) in the context of the AUTO-CAAS HöG project (number: 20140312).

[1]https://crest2016.inria.fr

stands for the set of all computations $\pi' = (s_0, l'_0, \mathcal{D}'_0), \ldots, (s_m, l'_m, \mathcal{D}'_m), s'_{m+1}$ such that $l'_0 \ldots l'_m$ is a sub-word of $l_0 \ldots l_n$.

We consider formulae in *Hennessy-Milner logic* (HML) [12] given by the following grammar:

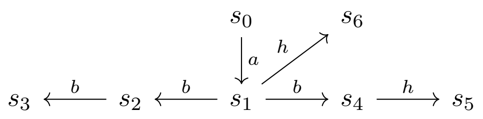
$$\phi, \psi ::= \top \mid \langle a \rangle \phi \mid [a]\phi \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \qquad (a \in A).$$

The associated satisfaction relation $\vDash$ is defined in the standard way, over states $s \in \mathbb{S}$ and HML formulae. We call formulae ϕ that hold in the initial state of a system *immediate effects.*

Defining causality. Our notion of causality complies with that of "actual causation" proposed in [11] and further adapted to the setting of concurrent systems in [15]. Consider a transition system $T = (\mathbb{S}, s_0, A, \rightarrow)$; *causal traces* for an HML property ϕ in T denoted by $\mathit{Causes}(\phi, T)$ is the set of all computations $\pi = (s_0, l_0, \mathcal{D}_0), \ldots, (s_n, l_n, \mathcal{D}_n), s_{n+1}$ such that

1. $s_0 \xrightarrow{l_0} \ldots s_n \xrightarrow{l_n} s_{n+1} \wedge s_{n+1} \vDash \phi$ *(Positive causality)*
2. $\exists \chi \in A^*, s' \in \mathbb{S} : s_0 \xrightarrow{\chi} s' \wedge s' \vDash \neg\phi$ *(Counter-factual)*
3. $\forall \chi' = l_0\chi_0 \ldots l_n\chi_n \in \{l_0 \ldots l_n\} \cup (A^* \setminus \mathit{traces}(\pi)),\ s' \in \mathbb{S} : s_0 \xrightarrow{\chi'} s' \Rightarrow s' \vDash \phi$ *(Occurrence)*
4. $\forall \chi' \in \mathit{traces}(\pi) \setminus \{l_0 \ldots l_n\},\ s' \in \mathbb{S} : s_0 \xrightarrow{\chi'} s' \Rightarrow s' \vDash \neg\phi$ *(Non-occurrence)*
5. $\forall \pi' \in \mathit{sub}(\pi) : \pi'$ *does not satisfy items 1. – 4. above (Minimality)*

Consider, for an example, the following LTS and the HML formula $\phi = \langle h \rangle \top$:


$$\begin{tikzcd}
 & & s_0 \arrow[d, "a"] & s_6 & \\
s_3 & s_2 \arrow[l, "b"'] & s_1 \arrow[l, "b"'] \arrow[ur, "h"] \arrow[r, "b"] & s_4 \arrow[r, "h"] & s_5
\end{tikzcd}$$

Item 1 above suggests that action a should be a cause for ϕ. Item 2 indicates that the hazard formula does not hold trivially everywhere as, for instance, $s_0 \xrightarrow{abb} s_3$ and $s_3 \nvDash \phi$. Item 3 states that ab cannot be considered a cause as it can non-deterministically lead to s_2 and s_3 and only $s_4 \vDash \phi$ holds. Item 4 states that $(s_0, a, [\varepsilon]), s_1$ is not a cause for ϕ because extending a with bb, for instance, violates ϕ and thereby violates item 3. However, $(s_0, a, [h, bb, bh]), s_1$ is a cause, because a leads to a hazard, all possible extensions of a with anything but h, bb or bh, the only ones being ε and b, also keep the hazard. Item 5 states that $(s_0, a, [\varepsilon, \varepsilon]), (s_1, b, [h, b]), s_4$ is not a cause because it is not minimal. This is because its sub-computation $(s_0, a, [h, bb, bh]),\ s_1$ is a cause as previously discussed.

(De-)composing causality. A *causal projection* of $T = (\mathbb{S}, s_0, A, \rightarrow)$ with respect to a property ϕ, is $T' = (\mathbb{S}', s_0, A, \rightarrow')$ such that $\mathbb{S}' = \{s_i \mid 0 \leq i \leq n{+}1 \wedge (s_0, l_0, \mathcal{D}_0), \ldots, (s_n, l_n, \mathcal{D}_n), s_{n+1} \in \mathit{Causes}(\phi, T)\}$ and $\rightarrow' = \{(s_i, l_i, s_{i+1}) \mid 0 \leq i \leq n \wedge (s_0, l_0, \mathcal{D}_0), \ldots, (s_n, l_n, \mathcal{D}_n), s_{n+1} \in \mathit{Causes}(\phi, T)\}$. We write $T \downarrow \phi$ to denote the causal projection of T with respect to ϕ. Intuitively, a causal projection is an LTS whose executions capture precisely all causal traces.

Theorem 1 and Theorem 2 below show that reasoning on causality with respect to disjunctions and, respectively, conjunctions of HML formulae in the context of interleaved LTS's can be reduced to reasoning on causality in the corresponding interleaved components. We consider standard notions of interleaving ($||$) and non-deterministic ($+$) choice between LTS's [18].

Theorem 1. Consider LTS's $T = (\mathbb{S}, s_0, A, \rightarrow)$ and $T' = (\mathbb{S}', s'_0, B, \rightarrow')$ such that $A \cap B = \emptyset$. Assume two HML formulae ϕ and ψ over A and B, respectively. Whenever ϕ and ψ are not immediate effects, the following holds: $T \mid\mid T' \downarrow (\phi \vee \psi) \ \simeq\ T \downarrow \phi + T' \downarrow \psi$.

Theorem 2. Consider $T = (\mathbb{S}, s_0, A, \rightarrow)$ and $T' = (\mathbb{S}', s'_0, B, \rightarrow')$ such that $A \cap B = \emptyset$. Assume two HML formulae ϕ and ψ over A and B, respectively. Whenever ϕ and ψ are not immediate effects, the following holds: $T \mid\mid T' \downarrow (\phi \wedge \psi) \ =\ (T \downarrow \phi) \mid\mid (T' \downarrow \psi)$.

Discussion. Counterfactual arguments have become the basis for a number of fault analysis, failure localization and software debugging techniques, such as delta debugging [23], nearest neighbor queries [20], counterexample explanation in model checking [10, 9] and why-because-analysis [13]. (De-)compositional verification has been studied in contexts such as model-checking [2, 4, 22] and model-based conformance testing [19, 21]. Our approach is based on our earlier work on decompositional verification of modal mu-calculus formulae [1]. Regarding compositional verification of causality, we are only aware of the line of work in [7, 5, 6, 8]. Our initial results only concern interleaving components, but our long-term vision is that modal decomposition will enable mechanized decomposition of the modal formula for communicating components, following the approach of [14, 1].

References

[1] L. Aceto, A. Birgisson, A. Ingolfsdottir, and M. Mousavi. Decompositional reasoning about the history of parallel processes. In *FSEN 2011*, volume 7141 of *LNCS*, pages 32–47. Springer, 2012.

[2] H. R. Andersen. Partial model checking (extended abstract). In *LICS*, pages 398–407, 1995.

[3] G. Caltais, S. Leue, and M. Mousavi. (De-)Composing Causality in Labeled Transition Systems. Technical Report soft-16-02, 2016.

[4] D. Giannakopoulou, C. S. Pasareanu, and H. Barringer. Component verification with automatically generated assumptions. *Autom. Softw. Eng.*, 12(3):297–320, 2005.

[5] G. Gößler and L. Astefanoaei. Blaming in component-based real-time systems. In *EMSOFT 2014*, pages 7:1–7:10. ACM Press, 2014.

[6] G. Gößler and D. Le Métayer. A general framework for blaming in component-based systems. *Sci. Comput. Program.*, 113:223–235, 2015.

[7] G. Gößler, D. Le Métayer, and J. Raclet. Causality analysis in contract violation. In *RV 2010*, volume 6418 of *LNCS*, pages 270–284. Springer, 2010.

[8] G. Gößler and J. Stefani. Fault ascription in concurrent systems. In *TGC*, volume 9533 of *LNCS*, pages 79–94. Springer, 2016.

[9] A. Groce, S. Chaki, D. Kroening, and O. Strichman. Error explanation with distance metrics. *STTT*, 8(3), 2006.

[10] A. Groce and W. Visser. What went wrong: Explaining counterexamples. In *SPIN*, LNCS 2648, pages 121–135. Springer, 2003.

[11] J. Halpern and J. Pearl. Causes and explanations: A structural-model approach. Part I: Causes. *The British Journal for the Philosophy of Science*, 2005.

[12] M. Hennessy and R. Milner. On observing nondeterminism and concurrency. In J. W. de Bakker and J. van Leeuwen, editors, *Automata, Languages and Programming, 1980, Proceedings*, volume 85 of *LNCS*, pages 299–309. Springer, 1980.

[13] P. Ladkin and K. Loer. Analysing aviation accidents using wb-analysis – an application of multimodal reasoning. In *AAAI Spring Symposium*. AAAI, 1998.

[14] K. G. Larsen and L. Xinxin. Compositionality through an operational semantics of contexts. *J. Log. Comput.*, 1(6):761–795, 1991.

[15] F. Leitner-Fischer and S. Leue. Causality checking for complex system models. In R. Giacobazzi, J. Berdine, and I. Mastroeni, editors, *VMCAI 2013, Rome, Italy, January 20-22, 2013. Proceedings*, volume 7737 of *LNCS*, pages 248–267. Springer, 2013.

[16] F. Leitner-Fischer and S. Leue. Probabilistic fault tree synthesis using causality computation. *International Journal of Critical Computer-Based Systems*, 4:pp. 119–143, 2013.

[17] D. Lewis. *Counterfactuals.* Blackwell Publishers, 1973.

[18] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS.* Springer, 1980.

[19] N. Noroozi, M. Mousavi, and T. Willemse. Decomposability in input output conformance testing. In *MBT 2013*, volume 111 of *Electr. Proc. in TCS*, pages 51–66, 2013.

[20] M. Renieris and S. Reiss. Fault localization with nearest neighbor queries. In *ASE*, Canada, 2003.

[21] T. Villa, N. Yevtushenko, R. Brayton, A. Mishchenko, A. Petrenko, and A. Sangiovanni-Vincentelli. *The Unknown Component Problem, Theory and Applications.* Springer, 2012.

[22] G. Xie and Z. Dang. Testing systems of concurrent black-boxes—an automata-theoretic and decompositional approach. In *FATES*, volume 3997 of *LNCS*, pages 170–186. Springer, 2006.

[23] A. Zeller. *Why Programs Fail: A Guide to Systematic Debugging.* Elsevier, 2009.