KaSa: A Static Analyzer for Kappa

Pierre Boutillier, Ferdinanda Camporesi, Jean Coquet, Jérôme Feret, Kim Quyên Lý, Nathalie Théret, Pierre Vignet

KaSa: a Static Analyzer for Kappa ⋆

Pierre Boutillier[1], Ferdinanda Camporesi[2], Jean Coquet[3], Jérôme Feret[2], Kim Quyên Lý[4], Nathalie Theret[5,6], and Pierre Vignet[6]

[1] Harvard Medical School, Boston, USA
Pierre_Boutillier@hms.harvard.edu
[2] DI-ÉNS (INRIA/ÉNS/CNRS/PSL⋆), Paris, France
camporesi@ens.fr, feret@ens.fr
[3] Department of Biomedical Data Science, Stanford University, Stanford, California
coquet.jean@gmail.com
[4] Tezos France
lykimq@gmail.com
[5] Univ Rennes, Inserm, EHESP, Irset - UMR_S1085, F-35043 Rennes, France
nathalie.theret@univ-rennes1.fr
[6] Univ Rennes, Inria, CNRS, IRISA F-35000 Rennes, France
pierre.vignet@inria.fr

Abstract. KaSa is a static analyzer for Kappa models. Its goal is two-fold. Firstly, KaSa assists the modeler by warning about potential issues in the model. Secondly, KaSa may provide useful properties to check that what is implemented is what the modeler has in mind and to provide a quick overview of the model for the people who have not written it.
The cornerstone of KaSa is a fix-point engine which detects some patterns that may never occur whatever the evolution of the system may be. From this, many useful information may be collected: KaSa warns about rules that may never be applied, about potential irreversible transformations of proteins (that may not be reverted even thanks to an arbitrary number of computation steps) and about the potential formation of unbounded molecular compounds. Lastly, KaSa detects potential influences (activation/inhibition relation) between rules.
In this paper, we illustrate the main features of KaSa on a model of the extracellular activation of the transforming growth factor, TGF-b.

Contribution. Jérôme Feret (2010-present) and Kim Quyên Lý (2015-2017) are the main contributors of KaSa. KaSa is integrated within the Kappa modeling platform whose main architect is Pierre Boutillier. In particular, Pierre Boutillier has integrated KaSa in the user interface of Kappa which may be used either

⋆ This material is based upon works partially sponsored by ANR (Chair of Excellence AbstractCell), the Defense Advanced Research Projects Agency (DARPA) and the U. S. Army Research Office under grant number W911NF-14-1-0367, and by the ITMO Plan Cancer 2014 (TGFSysBio project). The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of ANR, DARPA, the U. S. Department of Defense, or ITMO.

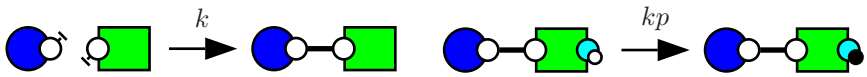

Fig. 1. Two rules. (left) Two proteins may bind. (right) The protein on the left may phosphorylate the right site of the protein on the right.

online, or locally. The model of the extracellular activation of the transforming growth factor, TGF-b, has been assembled by Jean Coquet (2012-2017), Nathalie Théret (2012-present), Pierre Vignet (2016-present), and Ferdinanda Camporesi (2016-present). Jérôme Feret has written the paper.

1 Introduction

Kappa may be used to describe systems of mechanistic interactions between proteins by the means of site-graph rewriting rules. Each node in graphs denotes an instance of a protein equipped with a kind and a finite set of identified sites. Rules may bind/unbind sites pair-wisely to establish/break links between proteins. Some sites may also have an internal state in order to specify if they are phosphorylated, methylated, and so on, so forth. We give, in Fig. 1, two examples of rules in Kappa.

Kappa is context-free: only the information that matters for a given interaction to happen has to be mentioned in rules. This feature is crucial to scale up to the size of large models. Thus Kappa provides the opportunity to design arbitrarily sophisticated models. These models may involve proteins with multiple phosphorylation sites, scaffolds, concurrency for shared resources, different time- and concentration scales, large variabilities in the kinds of molecular compounds, and non-linear feed-back loops. In general, we want to understand how the collective behavior of proteins may emerge from the mechanistic interactions between individual proteins. Yet, there are no modeling wizards and before investigating the long term behavior of a model, it is worth wondering whether the implementation matches faithfully our modeling assumptions. In the case of models written by others, extracting quickly some basic properties about models is also helpful to understand what the models are doing.

This motivates the use of formal methods. KaSa is a static analyzer that abstracts the set of reachable states of models, and then uses this information to collect insightful properties. In particular, KaSa may warn about rules that may never be applied, about potential definitive transformations of proteins and about the potential formation of unbounded molecular compounds. Lastly, KaSa detects the potential influences (activation/inhibition relation) between rules.

In this paper, we illustrate the main features of KaSa on a model of the extracellular activation of the transforming growth factor TGF-b, a protein which controls cell homeostasis in normal tissue, but promotes the development of fibrosis and cancer [6]. There has been a nice interplay between the design of the static analysis and the one of this model. On the first hand, KaSa has been helpful to curate the model, on the second hand, we have extended KaSa to

cope with new properties of interest that we have identified during the modeling process.

2 Technical description

Development. The development of KaSa has started in 2006, as a follow up of Complx, a static analyzer that had been designed by Plectix BioSystems (Cambridge, MA, USA). KaSa is now around 68,000 lines of OCaml [7] (excluding the front-end). It offers 53 command-line options. Jérôme Feret (2010-present) and Kim Quyên Lý (2015-2017) have been being the main developers.

Distribution. KaSa belongs to the Kappa modeling platform, which is completely open source `www.kappalanguage.org`. KaSa is partially integrated within the Kappa user interface, In particular, all the functionalities that are described in this paper, but local traces, are available on the fly while editing a model.

The development of the modeling platform is hosted on github `https://github.com/Kappa-Dev/KaSim`. An app is provided for MacOs and Windows. The nightly-builds of the development version may be downloaded at `https://tools.kappalanguage.org/nightly-builds/`. The modeling platform is also available as an opam package. With a properly installed opam, the instruction `opam pin add --dev KaSim` will compile all necessary dependencies as well as the current master branch of the git repository.

The manual may be consulted online at: `https://tools.kappalanguage.org/docs/KaSim-manual-master/KaSim_manual.htm` (see Chp. 6).

3 Main functionalities

Now we browse the main functionalities of KaSa.

Note that the results computed by KaSa depend all on the choice of the initial state, or more precisely on the set of the proteins and molecular compounds that may be present in the initial state independently of their concentration. KaSa is purely qualitative: its results depend neither on rule rates, nor on initial concentrations.

Reachability analysis. The cornerstone of KaSa is its reachability analysis. KaSa performs a mutual induction over some families of patterns, so as to prove that some of them may never occur in reachable states. Three families of patterns are considered [3]. The first one detects relations among the state of sites within each protein instance. The second one targets the relations between the state of sites in the proteins that are directly linked. The third one focuses on detecting whether or not a protein may be bound twice to the same instance of a protein. KaSa outputs a list of refinement lemmas. Each one consists in a precondition, that is a pattern, and a post-condition, that is a list of refinements of this pattern. The formal meaning of a refinement lemma is that whenever an instance of the

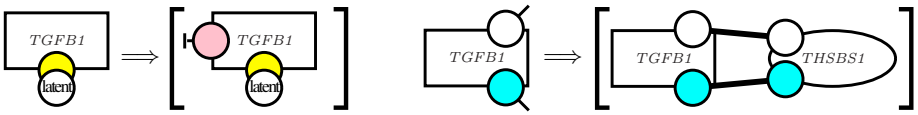

Fig. 2. Two refinement lemmas. (left) When TGFB1 is in its latent form, its site a is necessarily free. (right) When *TGFB1* has its two sites bound, it is bound twice to the same instance of the protein *THSBS1*. In each of these refinement lemmas, the refinement list is made of a single element. In more complicated cases, there maybe a choice of several patterns for refining the precondition.

precondition is found in a reachable state, this instance may be extended to an instance of a pattern in the post-condition.

In Fig. 2, we give some of the properties that are found in our case study. The analysis infers that in its latent form, *TGFB1* has always its site a (in pink) free. KaSa also detects that *TGFB1* may be bound twice to the same instance of the protein *THBS1*, but never to different instances simultaneously.

Dead rule detection. A rule the left hand side of which is in contradiction with the refinement lemmas cannot be applied whatever the evolution of the system is. There may be various reasons for this. Sometimes, several names have been used to denote the same protein. Sometimes, proteins have structural invariants that prevents the application of a rule. In our case study, dead rules have helped in identifying some missing parts in models, hence blocking the signaling pathways. The model has been completed after having consulted the literature.

Influences among rules. Rules may have a positive or a negative influence on each others. There is a positive (resp. negative) influence when an application of a given rule may potentially create (resp. remove) an instance of the left hand side of another rule. Influences provide an overview of the causality of the model.

We give an example in Fig. 3. We consider a protein with two phosphorylation sites. The left site may be freely phosphorylated and dephosphorylated, whereas the right site may get phosphorylated only when the left one is already phosphorylated. Thus the phosphorylation of the left site has a positive influence on the phosphorylation of the right one, while the dephosphorylation of the left site has a negative influence on the phosphorylation of the right one, as indicated in the influence map. This notion of influence is similar to the one that is used in Gene regulatory tools such as GinSIM [8] or reaction networks tools such as Biocham [2], except that, in Kappa, influences describe to which extent rules may influence each other, and not whether the variation of concentration of each molecular compound may influence the concentration of the other ones.

Since there are many rules, we use a hierarchy of abstractions to avoid the brute force approach which may not scale to large models. Firstly, we compute indirect influences. Indirect influences focus on the states of sites independently. There is a positive indirect influence whenever a rule may take a site into a state that is required by another rule to apply. Secondly, we compute direct influ-

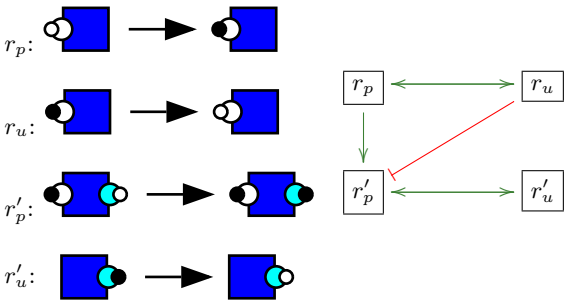

Fig. 3. Four rules (left) and the corresponding influence map (right). In rule r'_p, the right site may be phosphorylated only if the left site is. As a consequence the rule r_u inhibits the rule r'_p, and the rule r_p activates the rule r'_p.

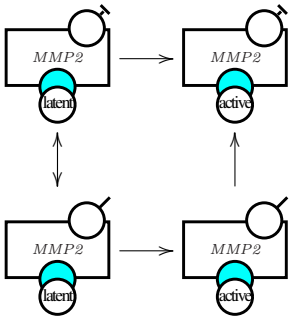

Fig. 4. A local trace. In protein *MMP2*, the passage in the active form is definitive and it prevents the site *CDB* to bind the protein *COL*.

ences. Direct influences are obtained by filtering indirect influences by checking that both rules have compatible requirements about their context of application. Thirdly, we refine direct influences further, by checking that the unifying context of both rules cannot be proved unreachable by our reachability analysis.

Local transition systems. It is sometimes useful to understand how a protein may go from one configuration to another. Thus KaSa computes a transition system for each kind of proteins of the model [4]. This abstraction completely ignores the context of the protein: the behavior of each protein is described independently without considering the state of the proteins it is attached to. Local traces do not intend to provide information about the collective behavior of proteins (i. e. their concentration): instead it focus on each protein individually.

Non weakly reversible transitions. Most mechanisms may be reverted in one or more steps of computation. This is crucial so that resources may be used several times (for instance an enzyme is expected to activate several instances of its substrate, thus it has to detach from it). However modelers often see signaling as cascades of interactions that push forward the signal.

Tarjan's strongly connected components decomposition algorithm is useful to detect which computation steps will never be reverted. Often, non weakly reversible transitions come from missing mechanisms and the model has to be completed. Sometimes, they come from a definitive degradation of a protein. In this case, the property is helpful to understand the behavior of this protein.

In the first versions of our case study, most rules about unbinding were missing. Our analysis has detected that corresponding binding steps were definitive and the model had to be completed accordingly. Once this done, all the remaining definitive transitions are related to the activation of the proteins *TGFB1*, *MMP2*, and *MMP14*, which is an irreversible process. In Fig. 4, we give a local trace associated to the protein *MMP2*.

model	Number of												Analysis time (second)								
	rules	constraints	dead rules	non weakly reversible transitions	rules with non weakly reversible transitions	indirect influences	direct influences	realisable influences	strongly connected components (Syntactic)	strongly connected components (Realisable)	edges per scc in average (Syntactic)	edges per scc in average (Realisable)	reachability analysis	influence map (Indirect)	influence map (Direct)	influence map (Realisable)	polymer detection (Syntactic)	polymer detection (Realisable)	non weakly reversible transitions	local traces	all-in-one (at maxiaml accuracy)
egfr_net	39	17	0	0	0	764	280	280	0	0	N/A	N/A	0.02	0.04	0.02	0.04	0.02	0.02	0.03	0.02	0.05
fceri_fyn	46	21	0	8	8	758	304	304	1	0	8	N/A	0.04	0.05	0.03	0.10	0.04	0.04	0.06	0.04	0.10
fceri_fyn_lig	48	21	0	8	8	760	306	306	1	0	8	N/A	0.04	0.05	0.03	0.09	0.04	0.05	0.06	0.05	0.10
fceri_fyn_trimer	362	22	36	96	96	59971	7405	6763	1	0	10	N/A	0.53	4.63	0.66	2.15	0.54	0.54	0.58	0.54	2.24
fceri_fyn_gamma2	59	21	0	0	0	1464	518	518	1	0	8	N/A	0.06	0.10	0.04	0.15	0.06	0.06	0.08	0.06	0.16
fceri_fyn_ji	36	16	0	0	0	536	231	231	1	0	8	N/A	0.03	0.02	0.01	0.06	0.03	0.03	0.05	0.03	0.07
fceri_fyn_lyn_745	40	18	2	2	2	620	255	243	1	0	8	N/A	0.04	0.04	0.02	0.07	0.04	0.04	0.05	0.04	0.08
fceri_fyn_trimer	192	19	0	0	0	21557	2536	2536	1	0	10	N/A	0.24	1.60	0.24	0.81	0.24	0.24	0.27	0.24	0.86
machine	220	72	7	17	10	5319	2873	2735	0	0	N/A	N/A	0.77	0.13	0.10	1.05	0.76	0.77	0.97	0.77	1.22
ensemble	233	86	0	1	1	4841	2936	2936	0	0	N/A	N/A	0.62	0.15	0.13	0.82	0.61	0.63	0.91	0.62	1.14
korkut (2017/01/13)	3916	1289	1610	2016	2016	75563	75563	39280	1	1	131	131	14	2.49	2.71	16	14	14	14	14	18
korkut (2017/02/06)	5750	2571	884	1397	1397	81412	75472	55101	1	1	2693	2687	94	4.01	4.16	94	96	115	99	114	119
TGF (V19)	97	107	10	153	53	3471	3009	2631	1	1	78	74	0.24	0.09	0.09	0.47	0.23	0.25	0.37	0.25	0.63
TGF (2018/04/19)	292	112	0	314	28	6040	5504	5504	1	1	108	108	0.89	0.18	0.19	1.36	0.86	0.90	1.25	0.92	1.73
BigWnt (2015/12/28)	356	134	1	833	14	5974	5271	5264	1	1	49	49	3.99	0.16	0.16	4.47	3.98	3.96	125	4.00	127
BigWnt (2017/03/22)	1486	182	12	61	16	1091187	38110	37958	1	1	84	80	15	26	5.15	25	15	15	260	15	286

Fig. 5. Benchmarks (performed on a MacBook Pro, 3.3 Ghz intel Core). For each model, we provide the number of rules, information about what has been discovered by KaSa and about analysis time.

Detection of unbounded polymers. Knowing which complexes may grow arbitrarily is important. Some models may assemble macro-molecules. But sometimes the presence of unbounded polymers is a side-effect of the lack of specification of the potential conflicts between protein interactions.

Unbounded polymers may only arise whenever a sequence of proteins may be repeated indefinitely in a reachable molecular compound. Such a sequence necessarily matches with a cycle in the oriented graph in which nodes are the different kinds of bonds between proteins (each kind bond is considered twice, one for each direction) and the edges connect two (oriented) bonds if the target of the first bond and the source of the second one are two different sites in a same kind of protein. We also use Tarjan's algorithm to detect these cycles.

This feature is available at two accuracy levels. At syntactic level, every kind of bonds occurring in the initial state or in the right hand side of a rule is considered. A more precise analysis is obtained by filtering out the pairs of bonds for which the corresponding pattern is proved unreachable.

In our case study, KaSa detects a large strongly collected component related to the formation of the Fibronectin matrix (which may indeed grow arbitrarily).

4 Benchmarks

We apply KaSa to several Kappa models (e. g. see Fig. 5). The first eight models are translations in Kappa of some of the models which are provided with the

BNGL distribution [1]. The models 'machine' and 'ensemble' are two versions of the MAPK signaling pathways published by Eric Deeds and Ryan Suderman [9]. Both versions of the model 'korkut' describe the Ras signaling pathways. They have been assembled by John Bachman and Benjamin Gyori (Sorger lab, Big-Mechanisms DARPA Project), following a three steps procedure [5]: automatic natural language processing, automatic assembling into Kappa, and human curation. We analyze two versions of the model of the extracellular matrix of the protein TGF-b that we have used to illustrate the different functionalities of KaSa. The assembling has been done by hand, by inspection of the literature and its curation has been assisted by KaSa. Lastly, we analyze two versions of the Wnt signaling pathway, written by Héctor F. Medina Abarca (Fontana Lab, Big-Mechanisms DARPA Project). This model has also been assembled by hand by inspection of the literature. Some scripts have been used to refine the kinetics of rules according to some contextual information about the proteins.

For each model, we give the number of constraints, of detected dead rules, of detected non weakly reversible transitions, of the rules that are involved in those transitions, of potential influence relation (in each accuracy mode), of strongly connected components that may occur in polymers (in both modes). Then, we give the CPU time used for each of these functionalities. The last column gives the CPU time of the whole analysis, with all functionalities set to the maximal accuracy level. It is worth noting that the CPU time required to compute the direct influence map, is sometimes longer than the one to compute the indirect one. Indeed, dumping an imprecise result may take longer than filtering this result thanks to a more costly but more accurate analysis.

References

1. Blinov, M., et al.: Bionetgen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. Bioinformatics 20(17) (2004)
2. Fages, F., Soliman, S.: From reaction models to influence graphs and back: A theorem. In: Fisher, J. (ed.) Proc. FMSB'08. Lecture Notes in Computer Science, vol. 5054, pp. 90–102. Springer (2008)
3. Feret, J., Lý, K.: Reachability analysis via orthogonal sets of patterns. ENTCS (2017), proc. SASB'16
4. Feret, J., Lý, K.: Local traces: an over-approximation of the behaviour of the proteins in rule-based models. IEEE/ACM TCBB (2018)
5. Gyori, B., et al.: From word models to executable models of signaling networks using automated assembly. bioRxiv (2017)
6. Horiguchi, M., Ota, M., Rifkin, D.: Matrix control of transforming growth factor-β function. The Journal of Biochemistry 152(4), 321–329 (2012)
7. Leroy, X., Doligez, D., Frisch, A., Garrigue, J., Rémy, D., Vouillon, J.: The ocaml system (2017), release 4.06
8. Naldi, A., Berenguier, D., Fauré, A., Lopez, F., Thieffry, D., Chaouiya, C.: Logical modelling of regulatory networks with ginsim 2.3. Biosystems 97(2), 134 – 139 (2009)
9. Suderman, R., Deeds, E.: Machines vs. ensembles: effective mapk signaling through heterogeneous sets of protein complexes. PLoS Computational Biology 9 (2013)