From word models to executable models of signaling networks using automated assembly

Benjamin M Gyori[1*], John A Bachman[1*], Kartik Subramanian[1], Jeremy L Muhlich[1], Lucian Galescu[2], Peter K Sorger[1]

[1] Laboratory of Systems Pharmacology, Harvard Medical School, Boston, USA

[2] Institute for Human and Machine Cognition, Pensacola, USA

[*] These authors contributed equally to this work

Address correspondence to:

Peter K Sorger

peter_sorger@hms.harvard.edu, cc: christopher_bird@hms.harvard.edu

Harvard Medical School, WAB438, 200 Longwood Avenue, Boston, MA, 02115

Tel: 617-432-6901/6902

Abstract

Word models describing molecular mechanisms are a common currency in spoken and written communication in biomedicine but are of limited use in predicting the behavior of complex biological networks. We present an approach to building computational models directly from natural language using automated assembly. Molecular mechanisms described in simple English are read by natural language processing algorithms, converted into an intermediate representation and assembled into executable or network models. We have implemented this approach in the Integrated Network and Dynamical Reasoning Assembler (INDRA), which draws on existing natural language processing systems as well as pathway information in Pathway Commons and other online resources. We demonstrate the use of INDRA and natural language to model three biological processes of increasing scope: (i) p53 dynamics in response to DNA damage, (ii) adaptive drug resistance in BRAF-V600E mutant melanomas, and (iii) the RAS signaling pathway. The use of natural language for modeling makes routine tasks more efficient for modeling practitioners and increases the accessibility and transparency of models for the broader biology community.

Keywords: computational modeling, natural language processing, signaling pathways

Running title: From word models to executable models

INTRODUCTION

Biophysics and biochemistry are the foundations of quantitative reasoning about biological mechanisms (Gunawardena, 2014a). Historically, biochemical systems were described in formal reaction diagrams and analyzed algebraically. As such systems became more complex and grew to include large networks in mammalian cells, word models (natural language descriptions), often accompanied by a pictogram or informal schematic, became the dominant way of describing biochemical processes. However it remains true that formal approaches involving dynamical models and systems theory can elucidate aspects of biological control that are obscured by informal descriptions. These include mechanisms such as all-or-none response to apoptosis-inducing ligands (Albeck *et al*, 2008; Rehm *et al*, 2002), sequential execution of cell cycle phases (Chen *et al*, 2004), the interplay of stochastic and deterministic reactions in the control of cell fate following DNA damage (Purvis *et al*, 2012), drug sensitivity and disease progression (Lindner *et al*, 2013; Fey *et al*, 2015), bacterial cell physiology (Karr *et al*, 2012) and similar biological processes. The challenge therefore arises of converting a plethora of word models in the literature into computational forms that can be rigorously analyzed. The technical environments used to create and explore dynamical models are unfamiliar to many biologists and there is little evidence that creation of new word models is being routinely supplemented by formal modeling approaches.

A variety of methods have been developed to make mechanistic modeling more powerful and efficient. These include fully integrated software environments (Loew & Schaff, 2001), graphical formalisms (Le Novère *et al*, 2009; Kolpakov *et al*, 2006), tabular formats (Tiger *et al*, 2012), high-level modular and rule-based languages (Smith *et al*, 2009; Mallavarapu *et al*, 2009; Danos *et al*, 2009), translation systems for generating SBML models from pathway information (Büchel *et al*, 2013; Ruebenacker *et al*, 2009) and specialized programming environments such as PySB (Lopez *et al*, 2013). These tools have increased the transparency and reusability of models, but have not sufficiently bridged the gap between verbal descriptions that dominate current literature and computational models.

To date, most attempts to make modeling more accessible have focused on graphical interfaces in which users draw reaction diagrams that are then used to generate equations. This approach is attractive in principle, since informal diagrams are a mainstay of most scientific presentations, but it has proven difficult in practice to accommodate the simultaneous demands of accurately rendering individual reactions while also depicting large numbers of interacting components. It is particularly difficult to create graphical interfaces that model the combinatorially complex reactions encountered in animal cell signaling (Stefan *et al*, 2014).

In this paper we explore the idea that natural language can serve as a direct input for dynamical modeling. Natural language has many benefits as a means of expressing mechanistic information: in addition to being familiar, it can concisely capture experimental findings about mechanisms that are ambiguous and incomplete. Extensive work has been performed on the use of software to convert text into computable representations of natural language, and natural language processing (NLP) tools are used extensively to mine the scientific literature (Fluck & Hofmann-Apitius, 2014; Krallinger *et al*, 2012). To our knowledge however, natural language has not been widely used as a direct input for computational modeling of biological or chemical processes. A handful of studies have explored the use of formal languages resembling natural language for model creation (Wasik *et al*, 2013; Kahramanoğullari *et al*, 2009) but these systems focus on capturing low-level reaction mechanisms and require that descriptions conform to a precisely defined syntax.

Three technical challenges are encountered in converting natural language models into executable models. The first is reading text with a machine in a manner that reliably identifies mechanistic assertions in the face of variation in how they are expressed. The second is designing an intermediate knowledge representation that captures often-ambiguous and incomplete mechanisms without adding unsubstantiated assumptions (thereby implementing the rule: "don't know, don't write"). This intermediate representation needs to be compatible with existing machine-readable sources of network information such as pathway databases. The third challenge is translating mechanistic assertions from the intermediate representation into executable models that involve varying mathematical formalisms and levels of detail; this involves supplying necessary assumptions left out of the original text.

The software tool described in this paper, the Integrated Network and Dynamical Reasoning Assembler (INDRA), addresses these challenges and makes it possible to construct different types of executable models directly from natural language and fragmentary information in pathway databases. In contrast to previous approaches to incorporating natural language in models, INDRA can accommodate flexibility in style and syntax through use of sophisticated NLP algorithms (Box 1). Mechanisms extracted from natural language and other sources are converted into *Statements* (the INDRA intermediate representation) and then translated into one of several types of models depending on the specific use case. We describe this process in some detail because it relates directly to how we understand and communicate biological mechanisms in papers and conversations. The fundamental challenges here are converting the informality and ambiguity of language, which is frequently a benefit in the face of incomplete information, into a precise set of statements (or equations) that constitute an executable mathematical model.

As a test case, we show that INDRA can be used to automatically construct a model of p53 dynamics in response to DNA damage from a few simple English statements; we show that the qualitative behavior of the INDRA model matches that of an existing mathematical model constructed by hand. In a second, more challenging test, we show that an ensemble of models of the MAP kinase pathway in melanoma cells can be built using literature-derived text describing the interaction of $BRAF^{V600E}$ and drugs used to treat the disease (Box 4). As a final test-case we use natural language and INDRA to assemble a large-scale model of the RAS pathway as defined by a community of RAS biology experts; we show how this model can be updated using a few simple sentences gathered from the RAS community .

Glossary

Application programming interface (API): a standardized interface by which one software system can use services provided by other software, often remotely; in the current context, INDRA accesses NLP systems and pathway databases via APIs. INDRA exposes an API that other software can build upon. API is used here interchangeably with *Interface* (e.g. INDRA's TRIPS *Interface*).

Molecular mechanism: used in this paper to refer to processes involved in changing the state of a molecular entity or in describing its interaction with another molecular entity as represented by a set of linked biochemical reactions. Mechanisms are often described in the literature and are captured in databases in formats such as BioPAX. The information we extract from such descriptions are interchangeably referred to as *mechanistic information, mechanistic assertions, mechanistic facts* and *mechanistic findings*.

Processor: a module in INDRA that constructs INDRA *Statements* from a specific input format.

Template extraction**:** the process by which INDRA *Processors* extract INDRA *Statements* from various input formats.

Assembler: a module in INDRA that constructs a model, network or other output from INDRA *Statements.*

Model assembly: the process of automatically generating a model in a given computational formalism from an intermediate knowledge representation; in our context from INDRA *Statements*.

Executable model**:** a computational model that can be simulated to reproduce the observable dynamical behavior of a system; often, but not always, a system of linked differential equations.

Policies: user-defined settings which affect the automated assembly process.

Knowledge representation: a formalism that allows aggregation of information, potentially from multiple sources, in a standardized computable format; in the current context, INDRA *Statements* serve as a common knowledge representation for mechanistic information.

***Natural language* (NL):** language that humans commonly use to communicate in speech and writing; in the current context, restricted to the English language.

***Natural language processing* (NLP)**: the algorithmic process by which a computer interprets natural language text.

***Named entity recognition* (NER)**: a sub-task of NLP concerned with the recognition of special words in a text that are not part of the general language; in the current context NER is used to identify proteins, metabolites, drugs, and other terms (which are generally referred to as *named entities*).

Grounding: a sub-task of NLP related to NER which assigns unique identifiers to named entities in text by linking them to ontologies and databases; in the current context this involves creating links to databases such as UniProt, HGNC, GO or ChEBI.

***Logical form (LF)*:** a graph representing the meaning of a sentence; an intermediate output of natural language processing in the TRIPS system (Box 1).

***Extraction knowledge base (EKB)*:** a collection of events and terms relevant to molecular biology that is the result of natural language processing with TRIPS (Box 1).

RESULTS

INDRA decouples the curation of mechanistic knowledge from model implementation

A core concept in INDRA is that the identification, extraction and regularization of mechanistic information (curation) is a distinct process from model assembly and implementation. Mechanistic models demand a concrete set of assumptions (about catalytic mechanisms, stoichiometry, rate constants, etc.) that are rarely expressed in a single paper or interaction extracted from a database. Models must therefore combine fairly generic assertions about mechanisms extracted from available knowledge sources (e.g., that enzyme E "activates" substrate S) with information or assumptions about molecular details (e.g., that the enzyme acts on the substrate S in a three-step ATP-dependent mechanism involving an activating site on the substrate) that derive from general knowledge about biophysical mechanisms. Precisely how such details are constructed depends on the requirements of the mathematical formalism, the specific biological use case, and the nature of the hypothesis being tested.

Text-to-model conversion in INDRA involves three coupled steps. First, text is processed into a machine-interpretable form and the identities of proteins, genes and other biological entities are *grounded* in reference databases. Second, the information is mapped onto an intermediate knowledge representation (INDRA *Statements*) that is designed to correspond in both specificity and ambiguity to descriptions of biochemistry as found in text (e.g. "*MEK1 phosphorylates ERK2")*. Third, the translation of this intermediate representation into concrete reaction patterns and then into executable forms such as networks of ordinary differential equations (ODEs) is performed in an *assembly* step. In this process, *Statements* capture mechanistic information available from the knowledge source without additions or assumptions, deferring interpretations of specific reaction chemistry that are often unresolved by the knowledge source but must be made concrete to assemble a model.

Information flow from natural language input to a model

The three steps involved in text-to-model conversion are implemented in a three-layer software architecture. An input layer comprising *Interface* and *Processor* modules (Figure 1A, block 1) is responsible for communicating with language processing systems (e.g., the TRIPS system) and pathway databases (e.g., the Pathway Commons database) to acquire information about mechanisms. An intermediate layer contains the library of *Statement* templates (Figure 1A, block 2), and a final output layer contains *Assembler* modules that translate *Statements* into formats such as networks of differential equations or protein-protein interaction graphs (Figure 1A, block 3). INDRA is written in Python and available under the open-source BSD license. Source code and documentation are available at http://indra.bio; documentation is also included as part of the Supplementary Information.

As an example of text being converted into an executable model, consider the sentence "*MEK1 phosphorylates ERK2 at threonine 185 and tyrosine 187.*" Figure 1B shows eight lines of Python code implementing this example; the numbers alongside each code block correspond to the three layers of the INDRA architecture in Figure 1A and implement the flow of information between the user, INDRA and external tools shown in Figure 1C. The user first enters the sentence to be processed and calls the *process_text* command in the INDRA TRIPS *Interface.* This function sends a request to the web service exposed by the TRIPS NLP system (Allen *et al*, 2015) (Figures 1B and 1C, block 1). INDRA can also call on the REACH NLP system, which has complementary capabilities (Valenzuela-Escarcega *et al*, 2015), but in this paper we focus exclusively on TRIPS. TRIPS parses the text into its *logical form* (Box

1, Supplementary Figure S1A), and then extracts mechanisms relevant to molecular biology into an extraction knowledge base (EKB; Box 1, Supplementary Figure S1B). Included in this process are entity recognition and grounding whereby MEK1 is recognized as a synonym of the HGNC gene name MAP2K1 and grounded to UniProt Q02750 (Erk2 is grounded to MAPK1 and UniProt P28482). These terms are explained in Box 1, in Supplementary Information section 2.1, and in (Allen *et al*, 2015). The TRIPS *Processor* in INDRA extracts *Statements* directly from the EKB output returned by TRIPS (Figures 1B and 1C, block 2). The translation of *Statement*s into concrete models is performed by an INDRA *Assembler*. In this example, a PySB *Assembler* was used to build a rule-based model in PySB (Lopez *et al*, 2013) and generate an SBML-compatible reaction network (Figures 1B and 1C, block 3). Because the Phosphorylation *Statements* in this example are compatible with multiple concrete reaction patterns, the user specifies a *policy* for assembly: here we used the "two-step" policy, which implements phosphorylation with reversible enzyme-substrate binding (polices are described below). The resulting reaction network was instantiated as a set of ODEs and simulated using default parameter values to produce the temporal dynamics of all three phosphorylated forms of ERK2 (labeled MAPK1; Figure 1C, bottom right). The same rule-based model can also be analyzed stochastically using network-free simulators (Danos *et al*, 2007b; Sneddon *et al*, 2011).

Box 1: Natural language processing using TRIPS

To convert text into computable representations that capture syntax and semantics INDRA uses external NLP software systems exposed as web services. This paper focuses on DRUM (Deep Reader for Understanding Mechanisms; http://trips.ihmc.us/parser/cgi/drum) which is a version of the general purpose TRIPS NLP system customized for extracting biological mechanisms from natural language text. TRIPS has been developed over a period of decades and has been used for natural language communication between humans and machines in medical advice systems, robotics, mission planning, etc. (see for example Ferguson & Allen, 1998; Chambers *et al*, 2005; Allen *et al*, 2006).

The first step in processing natural language with TRIPS is a "shallow" or syntactic analysis of the text to identify grammatical relationships among words in a sentence, recognize named entities such as proteins, amino acids, small molecules, cell lines, etc. and link these entities to appropriate database identifiers (the process of *grounding*). TRIPS then uses this information to perform a "deep" semantic analysis that tries to determine the meaning of a sentence in terms of its logical structure. This process draws on a general purpose semantic lexicon and ontology that defines a range of word senses and semantic relations among words. The output of this process is represented as a logical form (LF) graph (Manshadi *et al*, 2008). The LF graph represents the sense of each word (e.g. "protein") and captures the semantic roles of relevant arguments (e.g. "affected") for each predicate (e.g. "activation"). The LF also represents tense, modality and aspect information — information that is crucial for determining whether a statement expresses a stated fact, a conjecture or a possibility.

The LF graph is then transformed into an extraction knowledge base (EKB) containing extractions relevant for the domain, in this case molecular biology. LF graphs compactly represent and normalize much of the variation and complexity in sentence structure; EKBs can therefore be extracted from the LF using a relatively small set of rules. The EKB is an XML file containing entries for *terms* (e.g., proteins, drugs), *events* (e.g., activation, modification) involving those terms, and higher-level *causal relations* between the events. The EKB also contains additional information such as the text from which a given term or event was constructed.

A more thorough technical description of TRIPS/DRUM is given in Supplementary Information section 2.1 and in (Allen *et al*, 2015); a broader overview of NLP systems can be found in (Allen, 2003).

INDRA Statements represent mechanisms from multiple sources

INDRA *Statements* serve as the bridge between knowledge sources and assembled models and we therefore describe them in detail. *Statements* are implemented as a class hierarchy grouping related mechanisms (a UML diagram of existing *Statement* classes is shown in Supplementary Figure S2). Each INDRA *Statement* describes a mechanism involving one or more molecular entities, along with information specific to the mechanism and any supporting evidence drawn from knowledge sources. For example, the phosphorylation *Statement* shown schematically in Figure 2A contains references to enzyme and substrate *Agents* (which in this case refers to MAP2K1 and MAPK1, respectively), the phosphorylated residue and position on the substrate, and one or more *Evidence* objects with supporting information. An *Agent* is an INDRA object that captures the features of the molecular state necessary for a participant to take part in a molecular process (Figure 2B). This includes necessary post-translational modifications, bound co-factors, mutations, cellular location and state of activity (Figure 2B and Supplementary Figure S3). Agents also include annotations that *ground* molecular entities to unique identifiers in one or more databases or ontologies (e.g. HGNC, UniProt, ChEBI; Figure 2B). *Evidence* objects contain references to supporting text, citations and relevant experimental context (Figure 2C).

An important feature of both *Statements* and *Agents* is that they need not be fully specified. If there is no information in the source pertaining to a specific detail in a *Statement* or *Agent* then the corresponding entry is left blank; this is an example of the "don't know, don't write" principle. INDRA and the rule-based models it generates are designed to handle information that is incomplete in this way. For example, the phosphorylation *Statement* shown in Figure 2A indicates that the phosphorylation of substrate MAPK1 *can occur* when the enzyme MAP2K1 is phosphorylated at serine residues S218 and S222, but other aspects of the state of MAP2K1 are left unspecified (e.g., whether MAP2K1 is phosphorylated at S298, or bound to a scaffold protein such as KSR). *Statements* capture the ambiguity inherent in the vast majority of statements about biological processes thereby permitting multiple interpretations: for example, phosphorylation of MAP2K1 at S218 and S222 could be necessary and sufficient for activity against MAPK1, necessary but not sufficient, sufficient but not necessary, or neither sufficient nor necessary, depending on other molecular context outside the scope of the *Statement*. The ability of *Statements* to capture knowledge from input sources while making as few additional assumptions as possible is an essential feature of the text-to-model conversion process. It also conforms closely to the way individual experiments are described and interpreted since each experiment typically reveals only a subset of the knowledge needed to fully understand a biochemical mechanism or implement it in a model. The ambiguity in *Statements* is resolved during the *assembly* step in which assumptions are explicitly declared as a means to generate a well-defined executable model.

Normalized extraction of findings from diverse inputs using mechanistic templates

The principal technical challenge in extracting mechanisms from input sources is identifying and normalizing information contained in disparate formats (e.g., BEL, BioPAX, TRIPS EKB) into a common form that INDRA can use. To accomplish this, INDRA queries for patterns in the input formats corresponding to the existing *Statement* types (templates), matching individual pieces of information from the source format to fields in the *Statement* template. This procedure is implemented for each type of input, making it possible to extract knowledge in a consistent form. Template-matching does not guarantee that every mechanism found in a source can be captured by INDRA, but it does ensure that when a mechanism is recognized, the information is captured in a normalized way that enables downstream model assembly. The process is therefore configured for high precision at the cost of lower recall.

INDRA implements template-matching extraction for each input format using a set of *Processor* modules. In the case of natural language, the extraction knowledge base (EKB) output from TRIPS serves as an input for the TRIPS *Processor* in INDRA. For a statement such as "MAP2K1 that is phosphorylated on S218 and S222 phosphorylates MAPK1 at T185" the EKB extraction graph (Figure 3, top left) has a central node (red text) corresponding to a *phosphorylation* event that applies to the *terms* (blue text) MAP2K1 and MAPK1; the term "*threonine-185"* is a property of this event (green text depicts the grounding in UniProt and HGNC identifiers). A second *phosphorylation* event (yellow box) involving S218/S222 of MAP2K1 is recognized by TRIPS as a nested property of MAP2K1 phosphorylation. It is a precondition for the primary *phosphorylation* event on *MAPK1*.

INDRA establishes that this extraction graph corresponds to an INDRA Phosphorylation *Statement* and the template for such a *Statement* has entries for an enzyme, a substrate, a residue and a position (Figure 2A). The AGENT in the TRIPS EKB is identified as the enzyme which itself has a modification (phosphorylation) at specified positions (S218 and S222). The AFFECTED portion of the TRIPS EKB is identified as the substrate MAPK1. The extracted INDRA *Statement* collects this information along with target residue ("threonine") and position ("185") on the substrate. The end result is a biochemically plausible depiction of a specific type of reaction from a short fragment of text.

Extraction of a Phosphorylation *Statement* from databases using BioPAX or BEL follows the same general procedure. The INDRA BioPAX *Processor* uses graph patterns to search for reactions in which a substrate on the right hand side gains a phosphorylation modification relative to the left hand side (Figure 3, center left). The *Processor* identifies this as a phosphorylation reaction and constructs a Phosphorylation *Statement* for each such reaction that it finds.

In the case of BEL, statements consist of subject–predicate–object expressions describing the relationships between molecular entities or biological processes (Box 2). INDRA's *BEL Processor* queries a BEL corpus (formatted as an RDF graph) for expressions consistent with INDRA *Statement* templates. For example, Phosphorylation *Statements* are extracted by searching for expressions in which the subject represents the kinase activity of a protein that *directly increases* an object representing a modified protein (Figure 3, bottom left); *directly increases* is a predicate used when molecular entities interact physically. Triples that fit this pattern are extracted into an INDRA Phosphorylation *Statement* with the subject as the enzyme and the object as the substrate.

Box 2: BioPAX and BEL

BioPAX is a widely used format for describing biological interactions designed to facilitate the exchange and integration of pathway information from multiple sources (Demir *et al*, 2010). BioPAX is the core exchange format underlying the Pathway Commons database, which aggregates information from over 20 existing sources including Reactome, NCI-PID, KEGG, PhosphoSitePlus, BioGRID and Panther (Cerami *et al*, 2011). Pathway Commons provides a web service with an interface for submitting queries about pathways and recovering the result as a BioPAX graph; a query could involve finding all proteins and interactions in the neighborhood of a specified protein or finding all paths between two sets of proteins.

BioPAX employs a Web Ontology Language (OWL) knowledge representation centered around biochemical processes and reactants and is applicable to metabolic, signaling and gene regulatory pathways. The representation of reactions in BioPAX is flexible: an arbitrary set of complexes and standalone molecules on the left hand side of a reaction can produce complexes and molecules on the right hand side subject to one or more catalytic controllers.

The Biology Expression Language (BEL) was designed to facilitate the curation of knowledge from the literature in a machine-readable form. While BioPAX is designed to capture direct, molecular interactions, BEL can express indirect effects and higher level cellular- or organism-level processes; for example, one can represent the finding that the *abundance of BAD protein increases apoptosis*. Each BEL Statement records a scientific finding, such as the effect of a drug or other perturbation on an experimental measurement, along with contextual annotations such as organism, disease, tissue and cell type. BEL Statements are structured as subject, predicate, object (RDF) triples: the subject and object are BEL Terms identifying molecular entities or biological processes, and the predicate is a relationship such as *increases* or *decreases*. BEL has been used to create both public and private knowledge bases for machine reasoning; the BEL Large Corpus (see www.openbel.org) is currently the largest openly-accessible BEL knowledge base and consists of ~80,000 statements curated from >16,000 publications.

Assembly of alternative executable models from mechanistic findings

The role of INDRA *Assemblers* is to generate models from a set of *Statements*. This step is governed not only by the relevant biology, but also by the requirements of the target formalism and decisions about model complexity (e.g., the number of variables, parameters, or agents). INDRA has multiple *Assemblers* for different model formats; here we focus on the PySB *Assembler*, which creates rule-based models that can either be simulated stochastically or as networks of differential equations (Danos *et al*, 2007a; Faeder *et al*, 2009). Models assembled by INDRA's PySB *Assembler* can be exported into many widely used modeling formalisms such as SBML, MATLAB, BNGL and Kappa using existing PySB functions (Lopez *et al*, 2013).

Assembling an INDRA Phosphorylation *Statement* into executable form requires a concrete interpretation of information that is unspecified or ambiguous in the text or other source, a process we illustrate by describing three alternative ways to express the phosphorylation of MAPK1 by MAP2K1 (Figure 4). As a first step, assembly of the *Statement* requires a concrete interpretation of the partially specified state of the enzyme agent: MAP2K1 sites S218 and S222 are specified as being phosphorylated but no information is available about other sites or binding partners. In assembling rules, the PySB *Assembler* omits any unspecified context, exploiting the "don't care don't write" convention in rule-based modeling (Box 3) in which the states of unspecified sites are treated as being irrelevant for rule activity. The default interpretation is therefore that phosphorylation of MAP2K1 at S218 and S222 is *sufficient* for kinase activity; whether or not it is also *necessary* is determined by other rules involving MAP2K1 that may be in the model.

Box 3: Rule-based modeling and PySB

Accurate simulation of biochemical systems requires that every species be explicitly tracked through time. The combinatorial nature of protein complex assembly, post-translational modification and related processes causes the number of possible molecular states in many signaling networks to explode and exceed the capacity for efficient simulation (Stefan *et al*, 2014). For example, full enumeration of complexes involved in EGF signaling would require more than 10^{19} molecular species differing in their states of oligomerization, phosphorylation and adapter protein binding (Feret *et al*, 2009). Rule-based modeling (RBM) languages such as Kappa and BioNetGen (BNGL) address this challenge by allowing interactions among macromolecules to be defined using "rules" that specify the local context required for an molecular event to occur (Faeder *et al*, 2009; Danos *et al*, 2007a) . The molecular features that do not affect the event are omitted from the rule, a convention known as "don't care, don't write." Specifying molecular interactions in this way has two chief benefits: (i) it makes the representation of a model much more compact and transparent than a set of differential equations; (ii) it enables the simulation of very complex systems using network-free methods (Danos *et al*, 2007b). RBMs can also be translated into conventional modeling formalisms such as networks of ODEs.

Executable model assembly in INDRA is built on PySB, a software system that embeds a rule-based modeling language within Python to enable the use of macros and modules to concisely express recurring patterns such as catalysis, complex assembly, sub-pathways, etc. (Lopez *et al*, 2013). Rule-based modeling languages are well-suited to building executable models from high-level information sources such as natural language because assertions about mechanisms typically specify little molecular context. INDRA converts such assertions into one or more model rules using *policies* that control the level of detail.

The second step in the assembly of a Phosphorylation *Statement* is generating a concrete set of biochemical reactions that constitute an executable model. The challenge here is that the concept of protein "phosphorylation" can be realized in a model in multiple different ways. For example, a "*one-step*" policy converts an INDRA Phosphorylation *Statement* into a single bimolecular reaction in which a product (a phospho-protein) is produced in a single irreversible reaction without explicit consideration of enzyme-substrate complex formation (Figure 4, "one-step policy"; this produces one reaction rule and one free parameter). Such a representation is not biophysically realistic, since it does not reproduce behaviors such as enzyme saturation, but it has the advantage of requiring a single free parameter. One-step mechanisms are convenient for modeling coarse-grained dynamics and causal flows in complex signaling networks (Salazar & Höfer, 2006). A "two-step policy" is more realistic and creates two rules: one for enzyme-substrate binding and one for product release (Figure 4, "two-step policy"; two reaction rules and three free parameters). This is the most common interpretation of a phosphorylation reaction in existing dynamical models and correctly captures enzyme saturation, substrate depletion, and other important mass-action effects. However, the two-step policy does not explicitly consider ATP as a substrate, and cannot model the action of ATP-competitive kinase inhibitors at the enzyme active site. The "ATP-dependent" policy explicitly models the binding of ATP and substrate as separate reaction steps (Figure 4, "ATP-dependent policy;" three reaction rules and five free parameters). Other mechanistic interpretations of "phosphorylation" are also possible: for example, two-step or ATP-dependent policies in which the product inhibits the enzyme by staying bound (or re-binding) after the phospho-transfer reaction (Gunawardena, 2014b). Such rebinding can have a substantial impact on kinase activity.

At first glance, it might seem preferable to use the most biophysically realistic set of reactions in all cases. However, a fundamental tradeoff exists between model complexity and accuracy: as the biochemical representation becomes more detailed, the number of free parameters and intermediate species increases, reducing the identifiability of the model. Given such a tradeoff, the benefit of having multiple *assembly policies* becomes clear: alternative models can automatically be constructed from a single high-level biochemical assertion depending on their suitability for a particular modeling task. The transparency and repeatability of model generation using *assembly policies* is especially important for larger networks in which hundreds or thousands of distinct species are subject to adjustment as the biophysical interpretation changes.

To enable the simulation of reaction networks as ODEs in the absence of data on specific rate parameters, INDRA uses a set of biophysically plausible default parameters; for example, association rates are diffusion limited (10^6 $M^{-1}s^{-1}$), off-rates default to 10^{-1} s^{-1} (yielding a default K_D of 100 nM) and catalytic rates default to 100 s^{-1}. These parameter values can be adjusted manually or obtained by parameter estimation. An extensive literature and wide range of tools exist for parameter estimation using experimental data and they are directly applicable to models assembled by INDRA (Mendes & Kell, 1998; Moles *et al*, 2003; Eydgahi *et al*, 2013; Thomas *et al*, 2015). For simplicity, we do not discuss this important topic further and rely below either on INDRA default parameters or manually adjusted parameters (as listed in the Supplementary Information) to facilitate dynamical simulations.

Modeling alternative dynamical patterns of p53 activation

As an initial test of using INDRA to convert a word model and accompanying schematic into an executable model, we turned to a widely cited review in *Cell* that describes canonical patterns of how mammalian signaling systems respond to stimulus (Purvis & Lahav, 2013). Figure 5 of (Purvis & Lahav, 2013) depicts the dynamics of p53 response to single stranded and double stranded DNA breaks (SSBs and DSBs). Using a schematic illustration, Purvis and Lahav explained that pulsatile p53 dynamics arises in response to DSBs but sustained dynamics are induced by SSBs. The difference is attributed to negative feedback from the Wip1 phosphatase to the DNA damage sensing kinase ATM, but not to ATR. We wrote a set of simple declarative phrases (Figure 5B and C) corresponding to edges in the schematic diagram (Figure 5A) representing activating or inhibitory interactions between Mdm2 (an E3 ubiquitin-protein ligase), p53, Wip1 and ATM (or ATR) (yellow numbers in Figure 5A, B and C). We then used INDRA to read the text and assemble executable models in PySB that were instantiated as networks of ODEs and simulated numerically. For each model we plotted p53 activation over time using standard Python libraries (Oliphant, 2007).

We found that our initial set of phrases (the "word models" comprising sentences 1-5 in Figure 5B and sentences 1-6 in 5C) failed to reproduce the expected p53 dynamics for SSBs and DSBs: in our INDRA models SSBs induced steady, low-level activation of p53 and DSBs failed to induce oscillation (Supplementary Figure S5). One feature not explicitly included in the Purvis and Lahav diagrams and hence missing from our initial word models is the constitutive negative regulation of Mdm2 and Wip1. For clarity, visual representations of signaling pathways generally omit such inhibitory mechanisms despite their impact on dynamics (Heinrich *et al*, 2002). Of course, Purvis and Lahav were aware of these inhibitory reactions since these are found in their ODE-based dynamical models of p53 response to SSBs and DSBs (Batchelor *et al*, 2011). The specific reactions that inactivate Mdm2 involve binding by the catalytic inhibitor p14ARF (Agrawal *et al*, 2006) and inactivation of Wip1 is mediated by HIPK2-mediated phosphorylation and subsequent ubiquitin-dependent degradation (Choi *et al*, 2013) (depicted by dotted arrows and pink numbers in Figure 5A). We therefore added these reactions to the model as simple natural language phrases (denoted by pink numbers in 5B and C).

Following these changes, p53 response to SSBs correctly yielded sustained activation but the DSB response did not oscillate (Supplementary Figure S5). We noted that our DSB response model lacked a fundamental property of an oscillatory system, namely a time delay (Novák & Tyson, 2008). This delay was incorporated in the ODE model constructed by Lahav and colleagues (Batchelor *et al*, 2011) by using delay differential equations. Time delays can also be generated, however, by positive feedback (Novák & Tyson, 2008) and both ATM and ATR are known to undergo activating auto-phosphorylation (Bakkenist & Kastan, 2003; Liu *et al*, 2011). We therefore added the auto-activation of ATM or ATR to the text (denoted by dotted arrow and green numbers in Figure 5A, corresponding to green numbers in B and C). When assembled by INDRA, the extended word models resulted in p53 oscillation in response to DSBs (Figure 5C). Oscillation was robust to changes in kinetic parameters and initial conditions (Supplementary Table 2 and Supplementary Figure S5). In addition, the expanded model of ATR-driven p53 activation by SSBs still resulted in sustained p53 activation (Figure 5B, Supplementary Table 1 and Supplementary Figure S5). The key point in this exercise is that features essential for the operation of a dynamical system (e.g. degradation and auto-activation) were omitted from an informal diagram focusing on feedback for all the right reasons—brevity and clarity—but this has the unintended consequence of decoupling the informal representation from the dynamics being described. Concise machine-assembled word models are useful in this context because they ensure that descriptions and dynamical simulations are congruent.

The p53 model offers an opportunity to test how robust INDRA (and the TRIPS reading system) are to changes in the way input text is phrased. When we tested eight alternatives for the phrase "*Wip1 inactivates ATM*" ranging from "*Wip1 has been shown to deactivate ATM*" to "*ATM is inactivated by Wip1"* (Figure 5D, right, green sidebar) we found that all eight generated the same INDRA *Statement* and thus the same model as the original sentence. However, NLP is sensitive to spelling errors such as "*deaactivates"* [*sic*] and to grammatical errors such as "*Wip1 inactivate ATM*", and even some valid linguistic variants are not recognized (Figure 5D, right, red sidebar). We also tested whether differences in the way biological entities are named affects recognition and grounding; we found that *Wip1, WIP-1, WIP1, PPM1D* and *Protein phosphatase 1D* as well as *ATM, Atm* and *ataxia telangiectasia mutated* all worked as expected (Figure 5D, bottom, green). However, the recognition of protein and gene names in text is challenging and, for instance, "PP2C delta" was not recognized as a synonym for Wip1 (Figure 5D, bottom, red).

We then used INDRA to assemble a more detailed and mechanistically realistic model of p53 activation following DSBs (Figure 5E; POMI1.0). While the model in Figure 5C contained only generic activating and inhibitory reactions, the goal of POMI1.0 was to test the use of concepts such as phosphorylation, transcription, ubiquitination and degradation. We also used conditionals to describe the molecular state required for a protein to participate in a particular reaction (e.g. "*ubiquitinated p53 is degraded*"). The set of ten phrases shown in Figure 5E were assembled into 11 rules, 12 ODEs and 18 parameters (Supplementary Table 3). When we simulated the resulting ODE model we observed the expected oscillation in p53 activity (Figure 5E and Supplementary Figure S5). By adding and removing different phrases we found that including the mechanism "*Active ATM phosphorylates ATM*" was essential for oscillation; the phrase "*ATM phosphorylates itself*" generated a valid set of reactions but did not create oscillations for any of the parameter values we sampled. The difference is that "*Active ATM phosphorylates ATM*" corresponds to a trans-phosphorylation reaction—*i.e.* one molecule of ATM phosphorylates another molecule of ATM—which produces the non-linearity necessary for a time delay. In contrast, "*ATM phosphorylates itself*" represents modification in *cis,* which is incapable of generating oscillations in this example. It is known that ATM and ATR auto-phosphorylation occur in *trans* (Bakkenist & Kastan, 2003; Liu *et al*, 2011), validating this aspect of the model.

Taken together, these examples show how the process of creating a formal model from text highlights gaps in informal representations or descriptions of mechanisms. Introducing alternative assumptions and mechanisms using natural language is straightforward and can be accomplished by individuals with little or no technical expertise.

Modeling resistance to targeted therapy by vemurafenib

The MAPK/ERK signaling pathway is a key regulator of cell proliferation, differentiation and motility and is frequently dysregulated in human cancer (Box 4). Multiple ATP competitive and non-competitive (allosteric) inhibitors have been developed targeting kinases in this pathway. The most clinically significant drugs target RAF and MEK kinases in BRAF-mutant melanomas. For patients whose tumors express the oncogenic $BRAF^{V600E/K}$ mutation, treatment with the BRAF inhibitor vemurafenib (or, in more recent practice, a combination of the BRAF inhibitor dabrafenib and MEK inhibitor trametinib) results in dramatic tumor regression. Unfortunately, this is often followed by recurrence of drug-resistant disease 6 to 18 months later (Larkin *et al*, 2014). The mechanisms of drug resistance are under intensive study and include an adaptive response whereby MAPK signaling is reactivated in tumor cells despite continuous exposure to BRAF inhibitors (Shi *et al*, 2012a; Lito *et al*, 2012, 2013). Re-activation of MAPK signaling in drug-treated $BRAF^{V600E/K}$ cells is thought to involve disruption of ERK-mediated negative feedback (Figure 6A). The biochemistry of this process has been investigated in some detail and is subtle. For example, differential affinity of BRAF kinase inhibitors to monomeric and dimeric forms of BRAF are partly responsible for the ERK rebound (Kholodenko, 2015; Yao *et al*, 2015). However, the process within the scope of the MAPK signaling pathway has not been subjected to detailed kinetic modeling and several mechanistically distinct hypotheses have been advanced to describe the same drug adaptation phenomenon. Adaptation to BRAF inhibitors therefore represents a potentially valuable application of dynamical modeling to a rapidly moving field of cancer biology (Kholodenko, 2015).

We sought to use natural language to rapidly create models of MAPK signaling in melanoma cells using mechanisms drawn from the literature, with a particular focus on a series of highly influential papers from the Rosen lab (Joseph *et al*, 2010; Poulikakos *et al*, 2010; Lito *et al*, 2012; Yao *et al*, 2015). We also sought to establish whether natural language could be used to modify the resulting models so that different biochemical hypotheses could be tested by non-experts.

Box 4: The MAPK pathway and vemurafenib resistance in cancer

In normal cells, signal transduction via MAPK is initiated when an extracellular growth factor such as EGF induces dimerization of receptor tyrosine kinases (the EGFR RTK for example) on the cell surface. Dimerization and subsequent activation of RTKs results in assembly of signaling complexes at the plasma membrane and conversion of RAS-family proteins (HRAS, KRAS, and NRAS) to an active, GTP-bound state. RAS-GTP activates members of the RAF family of serine/threonine kinases (ARAF, BRAF, and RAF1), which serve as the first tier in a three-tier MAP kinase signaling cascade: RAF proteins phosphorylate MAP2K/MEK family proteins, which in turn phosphorylate the MAPK/ERK family proteins that control transcription factor activity, cell motility and other aspects of cell physiology. MAPK signaling is subject to regulation by feedback mechanisms that include inhibitory phosphorylation of EGFR and SOS by ERK, inhibition of the GRB2-mediated scaffold by SPRY family of proteins, and inhibition of ERK by DUSP proteins (Lito *et al*, 2012).

MAPK/ERK signaling is a key regulator of cell proliferation and is mutated in a variety of human cancers (Dhillon *et al*, 2007), with dramatic effects on cellular homeostasis. Overall, ~20% of all cancers carry driver mutations in one of the genes that encode MAPK pathway proteins (Stephen *et al*, 2014) and in the case of melanoma, 50% of cancers carry activating point mutations in BRAF (most commonly BRAF V600E). ATP-competitive inhibitors such as vemurafenib provide significant clinical benefit in treating BRAF mutant melanoma. However, remission of disease is transient, as tumors and tumor-derived cell lines develop resistance to vemurafenib over time (Lito *et al*, 2012). Recent studies have identified feedback regulation, bypass mechanisms, and other context-dependent factors responsible for restoring ERK signaling to pre-treatment levels (Shi *et al*, 2012b; Lito *et al*, 2012, 2013). For example, in the BRAF-V600E cell line A375, vemurafenib has been shown to suppress EGF-induced ERK phosphorylation completely upon treatment (Lito *et al*, 2013) but ERK phosphorylation levels rebound within 48 hours, with a concurrent increase in the level of RAS-GTP, the active form of RAS (Lito *et al*, 2012). It is the biology of this adaptation that we aim to capture in an INDRA model.

The baseline MAPK model (Melanoma ERK Model in INDRA; MEMI1.0) consists of 14 sentences describing canonical reactions involved in ERK activation by growth factors (Figure 6B, MEMI1.0) and corresponds in scope to previously described models of MAPK signaling (Stites *et al*, 2007; Birtwistle *et al*, 2007). In the baseline model, BRAF^{V600E} constitutively phosphorylates MEK as long as it is not bound to vemurafenib (sentence 9: "*BRAF V600E that is not bound to Vemurafenib phosphorylates MEK*"). A two-step policy that involves reversible substrate binding was used to assemble all phosphorylation and dephosphorylation reactions. For simplicity, we did not specify residue numbers or capture multi-site phosphorylation, instead modeling each step in the MAPK cascades as a single, activating phosphorylation event. With these assumptions, 14 sentences were processed by TRIPS to yield 14 INDRA Statements that were assembled into 28 PySB rules and 99 differential equations; the network of coupled ODEs was then simulated.

A key property of vemurafenib-treated $BRAF^{V600E}$ cells as described by Lito *et al*. is that the drug initially reduces pERK below the steady state level but pERK then rebounds despite the continued presence of vemurafenib. Levels of RAS-GTP (the active form of RAS) also increase during the rebound phase (Lito *et al*, 2012). In MEMI1.0, addition of EGF causes activation of RAS and phosphorylation of ERK at steady state. Addition of vemurafenib rapidly reduces pERK levels (Figure 6B) but extended simulations under a range of EGF and vemurafenib concentrations show that the amount of active RAS depends only on the amount EGF and is insensitive to the amount of vemurafenib; moreover, no rebound in pERK is observed in the presence of vemurafenib (Figure 6B and Supplementary Figure S6A). Thus, MEMI1.0 fails to capture drug adaptation.

In a series of siRNA-mediated knockdown experiments Lito *et al*. demonstrated that the pERK rebound involves an ERK-mediated negative feedback on one or more upstream regulators such as Sprouty proteins (SPRY), SOS or EGFR. To identify a specific mechanism that might be involved we used the BioPAX and BEL search capabilities built into INDRA. We queried Pathway Commons (Cerami *et al*, 2011) for BioPAX reaction paths leading from ERK (MAPK1 or MAPK3) to SOS (SOS1 or SOS2) and obtained multiple INDRA *Statements* for a MAPK1 phosphorylation reaction that had one or more residues on SOS1 as a substrate (including SOS1 sites S1132, S1167, S1178, S1193 and S1197). However, Pathway Commons did not provide any information on the effects of these phosphorylation events on SOS activity. To search for this we used the BEL *Interface* in INDRA to query the BEL Large Corpus (Catlett *et al*, 2013, Box 2) for all curated mechanisms directly involving SOS1 and SOS2. We found evidence that ERK phosphorylates SOS and that ERK inactivates SOS (Corbalan-Garcia *et al*, 1996). We did not find a precise statement in either database stating that phosphorylation of SOS inactivates it, but upon further investigation the publication referred to as evidence in (Corbalan-Garcia *et al*, 1996) describes a mechanism whereby SOS phosphorylation interferes with its association with the upstream adaptor protein GRB2. To include the inhibitory phosphorylation of SOS by ERK we therefore modified three sentences (Figure 6C, Model 2, Sentences 4, 5, and 14) in Model 1 and added two new sentences (Figure 6C, Model 2, sentences 15 and 16).

Though INDRA can assemble *Statements* derived from databases directly into models, in this case human curation (via changes to the natural language text) was required to identify gaps in the mechanisms available from existing sources.

The inclusion of SOS-mediated feedback in the model resulted in 16 declarative sentences that were translated into a MEMI1.1 model having 34 rules and 275 ODEs. Assembly of MEMI1.1 involved imposing assumptions that limited combinatorial complexity. For instance, in sentence 15 (Figure 6C) we specified that ERK cannot be bound to DUSP6 for ERK to phosphorylate SOS. While it is not known whether or not ERK can bind both DUSP6 and SOS at the same time, allowing for this possibility would introduce a "combinatorial explosion" (Faeder *et al*, 2005; Feret *et al*, 2009) in the number of reactions and make mass-action simulations challenging. It is common to make simplifying assumptions of this type in dynamical models (see for instance (Chen *et al*, 2009)), and an advantage of using natural language is that the assumptions are clearly stated. When MEMI1.1 was simulated we observed that, given a sufficient level of basal activity by addition of EGF, addition of vemurafenib resulted in dose-dependent increases in active RAS over pre-treatment levels (Supplementary Figure S6B). However, pERK levels remained low, suggesting that negative feedback alone (at least as modeled in MEMI1.1) is insufficient to explain the rebound phenomenon observed by Lito *et al*. (Figure 6C, Supplementary Figure S6B).

It has been suggested that RAF dimerization plays an important role in cellular responses to RAF inhibitors (Lavoie *et al*, 2013; Yao *et al*, 2015). Both wild-type and $BRAF^{V600E}$ dimers have a lower affinity for vemurafenib as compared to their monomeric forms (Yao *et al*, 2015). Moreover, Lito *et al*. observed that the reactivation of ERK following vemurafenib treatment was coincident with formation of RAF dimers, leading to the suggestion that vemurafenib-insensitive dimers in cells play a role in the re-activation of ERK signaling (Kholodenko, 2015). To model this possibility, we created MEMI1.2 in which binding of vemurafenib to monomeric or dimeric BRAF is explicitly specified by separate sentences, allowing the effects of different binding affinities to be explored (Figure 6D). Assembly of this model yielded 353 ODEs, many of which were accounted for by the combinatorial complexity of BRAF dimerization and vemurafenib binding (Supplementary Figure S7). Simulation showed that RAS activation increases and settles at a higher level following vemurafenib treatment, with the magnitude of the increase dependent on the amount of EGF and the concentration of drug (Figure 6D, Supplementary Figure S6C). Following a period of pERK suppression, a rebound in activity to ~30% of the fully active level is observed (Figure 6D) effectively recapturing the key findings of Lito *et al*. Subsequent work has shown that resistance to vemurafenib can also involve proteins such as DUSP, SPRY2 (Lito *et al*, 2013) and CRAF (Montagut *et al*, 2008). These mechanisms do not feature in the models described here, but can be included in MEMI by adding a few phrases to the word model.

This example demonstrates that it is possible to use INDRA to model signaling systems of practical interest at a scope and level of detail at which interesting biological hypotheses can be explored and tested. Comparison of models MEMI1.0 to 1.2 suggests that both feedback and BRAF dimerization are necessary for vemurafenib adaption and pERK rebound, in line with experimental evidence. The number of free parameters in these models varies, and we have not performed formal model calibration or verification, so the conclusion that MEM1.2 is superior to 1.0 is not rigorously proven. However, INDRA-assembled rule sets represent a solid starting point for modeling that involves rigorous comparison to data.

One issue we encountered in assembling these models was controlling the combinatorial complexity arising from the formation of protein complexes from a single set of reactants. This is a known challenge in dynamical modeling of biochemical networks with poorly understood implications for cellular biochemistry (Faeder *et al*, 2005; Harmer *et al*, 2010; Sneddon *et al*, 2011). From the perspective of an INDRA user, it is likely to manifest itself as a problem that can only be diagnosed at the level of PySB rules or ODE networks. We will need to develop new software systems to help non-technical users deal with such problems. Until then, it should be possible for non-expert users to modify most if not all sentences in a complex INDRA model as a means to explore alternative reaction mechanisms.

An extensible and executable map of the RAS signaling pathway

The BRAF pathway described above is part of a larger immediate-early signal transduction network with multiple receptors as inputs and transcription, cell motility and cell fate determination as outputs. RAS is a central node in this network and is an important oncogenic driver (Stephen *et al*, 2014). The ubiquity of RAS mutations in cancer has led to renewed efforts to target oncogenic RAS and RAS effectors. As a resource for the community of RAS researchers, the NCI RAS Initiative has created a curated pathway diagram that defines the scope of the RAS pathway as commonly understood by a community of experts (Stephen *et al*, 2014). Such pathway diagrams can serve as useful summaries, but unless they are backed by an underlying computable knowledge representation they are of limited use in quantitative data analysis.

We used INDRA to describe the RAS signaling network and automatically generated a diagram (Figure 7A, right) corresponding to the community-curated Ras Pathway v1.0 diagram (available at http://www.cancer.gov/research/key-initiatives/ras/ras-central/blog/what-do-we-mean-ras-pathway). We described the interactions in natural language (Figure 7A left, full text shown in Supplementary Information section 2.4) and used the TRIPS reading system to convert the description into INDRA *Statements*. A node-edge graph was generated using INDRA's *Graph Assembler* and rendered by the Graphviz software (Figure 7A, right). The pathway map is visually comparable to one drawn by hand and allows natural language-based editing and extension of the underlying set of mechanisms. After the v1.0 RAS diagram was distributed, the diagram's creators solicited verbal feedback from a large number of RAS biologists both in person and via a discussion forum. Suggestions from the community consisted of corrections and extensions. Using INDRA, these revisions can be made directly by editing the natural language source material. For example, one contributor noted that in the published pathway diagram (Figure 7A, right), P90RSK is activated by the mTORC2 complex, whereas in fact it is a substrate of MAPK1 and MAPK3 (https://www.cancer.gov/research/key-initiatives/ras/ras-central/blog/2014/what-do-we-mean-ras-pathway#comment-1693526648). We modified the natural language description to reflect this correction by removing the sentence "*mTORC2 activates P90RSK*" and replacing it with "*MAPK1 and MAPK3 activate P90RSK*." The pathway map obtained after assembly from the revised text correctly reflects the change (Figure 7B).

Several readers also suggested expansion of the pathway map to include other relevant proteins. Extensions of this type are easy to achieve using natural language: for example, we extended the v1.0 RAS diagram to include *JNK*, a MAP kinase that is induced in most cells by cytokines and stress (Anafi *et al*, 1997; Antonyak *et al*, 1998; Wagner & Nebreda, 2009). This was achieved by adding four sentences (Figure 7C, top) including "*MAP3K7 activates MKK4 and MKK7*" and "*MKK4 and MKK7 activate JNK1 and JNK2*". The subnetwork appended to the diagram is shown in Figure 7C (bottom). Note that we used common names for the JNK pathway kinases in the word model but INDRA canonicalized these to their official gene names (e.g., "HPK1", "MKK4", and "JNK1" were converted to MAP4K1, MAP2K4, and MAPK8, respectively).

The set of mechanisms used to generate the diagrams in Figures 7A-C can also be translated into a qualitative predictive model. We used the Simple Interaction Format (*SIF*) *Assembler* in INDRA to generate a Boolean network corresponding to the natural language pathway description in Figure 7A (see Supplementary Information section 2.4 for the rules comprising the network). Using such a Boolean network we can interpret signaling data and make predictions about perturbations. For example, we simulated the effects of adding growth factors and MEK inhibitors on phosphorylated c-Jun. The Boolean network simulation correctly predicted that c-Jun would be phosphorylated in the presence and absence of MEK inhibition (Figure 7D, blue). We then instantiated the extended network in Figure 7C (which identifies the JNK pathway as a possible contributor to c-Jun phosphorylation). In this case joint inhibition of JNK and MEK was required to fully inhibit c-Jun phosphorylation (Figure 7D, green). The biology in this example is relatively straightforward but it demonstrates that natural language descriptions of mechanisms, along with automated assembly into executable forms, can be used as an efficient and transparent way of creating extensible knowledge resources for data visualization and analysis.

DISCUSSION

In this paper we described a software system, INDRA, for constructing executable models of signal transduction directly from text. The process involves using natural language reading tools (TRIPS, in this paper) to convert text into a computer-intelligible form, identifying biochemical mechanisms and casting them in an intermediate knowledge representation that is decoupled from both input and output formats. The intermediate representation, comprising a library of INDRA *Statements*, is then used to assemble computational models of different types including networks of ODEs, Boolean networks, and interaction graphs according to user-specified policies that determine the level of biophysical detail. We applied the approach to three successively more ambitious use cases: (i) translating a diagram and accompanying text describing p53 regulation by DNA damage, (ii) modeling adaptive drug resistance in BRAF^{V600E} melanoma cells exposed to the BRAF inhibitor vemurafenib and (iii) constructing a large-scale model of RAS-mediated immediate-early signaling based on a crowd-sourced schematic drawing. These examples demonstrate the surprising ability of machines to exploit the flexibility and ambiguity of natural language while also adding prior knowledge about reaction mechanisms to create well-defined executable models.

The p53 POMI model represents a case in which a non-expert INDRA user should be capable of building a model from scratch and then editing and updating the model to explore alternative hypotheses. We based POMI on a word model found in a highly cited review but found it necessary to add certain mechanisms to reproduce the described oscillations in p53 (i.e., constitutive degradation and dephosphorylation steps and a positive feedback step involving auto-phosphorylation of ATM in *trans*). This example highlights the potential of natural language to expose important and frequently overlooked differences between a formal representation of a mechanism (in this case, a network for ODEs) and a diagram that purports to describe it. Direct conversion of text into models via INDRA helps to minimize such mismatches while keeping the description in an accessible and easily editable natural language form.

Construction of the BRAF^{V600E} MEMI model in INDRA is more involved, because it is necessary to control combinatorial complexity through dexterous use of language; currently, unwanted model complexity can best be diagnosed at the level of rules and equations, which requires some expertise in computational biology. However it should be possible to develop semi-automated means for diagnosing and correcting such problems. The RAS pathway is the most complex network tackled in this paper, but by restricting the mechanisms to positive and negative regulation and binding it remains manageable. Such a model could in principle be solicited directly from the community and we plan to release the INDRA RAS model to the same group of experts that helped Frank McCormick and colleagues build and improve the original RAS schematic (Stephen *et al*, 2014).

Challenges in generating executable models from text

Automating the construction of detailed biochemical models from text involves overcoming three technical challenges. The first is turning text into a computable form that correctly captures the biochemical events described in a sentence (typically verbs or actions) and the precise biomolecules involved (typically the subjects and objects of a phrase or sentence). This is possible in our system because TRIPS can extract meaning from sentences describing complex causal relationships in the face of variations in syntax (Box 1). TRIPS performs an initial shallow syntactic search to identify and ground named entities (genes, proteins, drugs, etc.) and then uses biology-specific ontologies to perform "deep" or semantic analysis, determining the meaning of a sentence in terms of its logical structure.

The second challenge involves extracting and normalizing information about mechanisms contained in NLP output. INDRA extracts mechanistic information from graphs generated by TRIPS by searching for matches to a predefined set of templates corresponding to biochemical processes (e.g., phosphorylation, transcription, binding, activation, etc.; Figure 3). These templates regularize the description of biochemistry in text by capturing relevant information in pre-determined fields: for example, a template for phosphorylation is structured to have a protein kinase, a phosphorylated substrate, and a target site. Information extracted by this template matching procedure is stored in corresponding fields in INDRA's intermediate representation, *Statements;* missing fields are left blank. INDRA *Statements* currently encompass terms and reactions commonly found in signal transduction pathways and gene regulation; however, the system is being extended to include a wider variety of biochemical processes.

The third challenge in text-to-model conversion is constructing an executable model from high-level mechanistic facts acquired from input sources. Knowledge of reaction type and reactant identity is insufficient to construct a detailed biophysical model: additional information derived from an understanding of the underlying biophysics is almost always required. For example, the conversion of a phosphorylation *Statement* into a reaction network can involve one-step kinetics, reversible two-step kinetics or two-step kinetics with explicit ATP binding. Conversion of *Statements* into explicit models is controlled by the imposition of *assembly policies* (Figure 4). Greater biophysical realism comes at the cost of increased model complexity and reduced parameter identifiability. Thus, there is no single optimal approach to model instantiation: the level of detail is determined by the purpose of the model and the way it is formulated mathematically.

Perhaps unexpectedly, constructing executable models from pathway databases using BioPAX or BEL presents similar challenges as constructing models from text. BioPAX and BEL statements are more structured than natural language, but they too lack the specificity needed to create executable models (Ruebenacker *et al*, 2009; Büchel *et al*, 2013). We therefore subject pathway information from databases to an analogous process as text, using templates and assembly policies to control the generation of specific reaction patterns.

Separating Model Content and Implementation

Most approaches to modeling biological networks directly couple the specification of scope and the collection of relevant facts to mathematical implementation. For example, in an ODE-based model, molecular species are directly instantiated as variables and related to each other using one or more differential equations for each mass action reaction (Figure 8A “Ordinary differential equations” and Figure 8B, left). Although conceptually straightforward, the lack of division between content and implementation makes it difficult to update a model with new content (e.g., new findings from the literature or new hypotheses), to change the level of biophysical detail or to switch mathematical formalisms. Programmatic modeling overcomes some of these problems by allowing the construction of models at a higher level of abstraction in which users implement reusable and composable macros and modules (Figure 8A “PySB Macro” and Figure 8B, center) (Lopez *et al*, 2013; Mallavarapu *et al*, 2009; Smith *et al*, 2009). The mathematical equations necessary for simulation are then generated automatically from the abstract representations.

INDRA introduces a further level of abstraction whereby a user describes a set of reactions in natural language or searches for related mechanisms in pathway databases and then uses a machine to turn these facts into executable models (Figure 8A "Natural language" and Figure 8B, right). In this process a user has full control over the *content* of the model and the level of detail, as specified by policies, but model *assembly* happens automatically. This decoupling simplifies the creation of dynamical models from natural language descriptions, enables the creation of models differing in detail or mathematical formalism and makes sure that verbal and mathematical descriptions of the same process are in correspondence (Figure 8B, right).

The decoupling of biological knowledge from specific applications reflects the way in which biologists work intuitively. We acquire informal knowledge through years of reading and experience, but this knowledge remains highly flexible; it allows for uncertainty about particular details and can be applied to a diverse set of problems in the lab. The ambiguity inherent in verbal descriptions of mechanisms conforms closely to the way in which individual experiments are designed and interpreted: it is extremely rare for one experiment to elucidate the status of all relevant sites of post translational modification, regulatory subunit binding or allosteric regulation of an enzyme. Natural language allows biologists to communicate this kind of provisional knowledge without prematurely resolving ambiguities or presupposing the biological context or experimental format in which the knowledge might be applied. In this respect INDRA mirrors the way biologists gather mechanistic information and apply it to specific research questions.

Relationship to previous work

Several tools have been developed to partially automate the construction of executable models from bioinformatics databases such as KEGG, Pathway Commons, and others (Ruebenacker *et al*, 2009; Wrzodek *et al*, 2013; Büchel *et al*, 2013; Turei *et al*, 2016a). Automating model translation in this way increases throughput and maintains links between model assumptions and curated findings in databases, eliminating the need for labor-intensive annotations of hand-built models (Le Novère *et al*, 2005). Such approaches have been particularly successful in the field of metabolism in which knowledge about enzyme-substrate reactions is well curated and closely corresponds in level of detail to what is required for mechanistic modeling (Büchel *et al*, 2013). In signal transduction, curation is less complete, the number of molecular states and interactions is far higher and networks vary dramatically from one cell type to the next. This complexity has been addressed for the most part by using strictly qualitative formalisms that describe positive and negative influences between nodes (Büchel *et al*, 2013; Turei *et al*, 2016b). In contrast INDRA uses an intermediate representation that encompasses both mechanistic processes (e.g., phosphorylation) and empirical causal influences (e.g., activation and inhibition). The model assembly procedure makes use of mechanistic information where available, but can incorporate qualitative influence relationships when mechanisms are not known.

Tools for assembling executable signaling models from pathway information can also be distinguished by the relationship between the number of input and output formats supported. The first instance of a software system for converting formats allowed one-to-one conversion from BioPAX to SBML (Ruebenacker *et al*, 2009). More recent one-to-many tools translate information from a single knowledge source into multiple output formats (Wrzodek *et al*, 2013), while many-to-one tools aggregate pathway information from many sources but target a single output format (Turei *et al*, 2016a). Use of an intermediate representation allows INDRA to decouple input and output formats and perform many-to-many conversion involving text, BioPAX, BEL, PySB, BNGL, SBML, ODEs, logical models and graph-based formats.

Limitations and future extensions of INDRA

INDRA focuses exclusively on the construction and revision aspects of a modeling project in which the goal is to gather information about relevant mechanisms and specify model structure. By design, the software does not perform parameter estimation, simulation or model analysis, leaving these tasks to other tools and methods. INDRA is likely to be most useful in facilitating collaboration between biologists with domain-specific expertise and computational biologists. The advantage of natural language in this context is that it makes it easy for teams to communicate about biological hypotheses and mechanisms without becoming mired in details of model implementation.

Limitations of model building with INDRA can be grouped into two categories: issues relating to the external NLP systems and the internal representation and assembly systems. In this paper we construct models using simple declarative sentences that lack much of the complexity and ambiguity of spoken language or the scientific literature. Declarative language can express a wide variety of biological mechanisms at different levels of detail and ambiguity and its primary strength is that it mitigates many of the difficulties associated with NLP-based extraction of biological mechanisms. Although TRIPS and INDRA are robust to variation in syntax and naming conventions, they cannot understand all possible ways a concept can be stated; for example "*Wip1 makes ATM inactive*" is not recognized as a substitute for "*Wip1 inactivates ATM*" (Figure 5). In such cases rephrasing is usually successful. The TRIPS system (as well as other NLP systems we tried, such as REACH) *can* be used to process the more complex and ambiguous language used in scientific papers, but the results are less robust due to the greater technical and conceptual challenges involved. Elsewhere we will describe progress on the task of extracting pathway information from the literature, which presents challenges not only for NLP but also for assembly (due to the large amount of irrelevant, redundant, overlapping, and erroneous information returned). In the approach described here, human domain experts simplify both the NLP and assembly challenges by digesting complex biological descriptions and summarizing them in simplified language.

The domains of knowledge covered by INDRA are currently limited by the scope of the intermediate representation and assembly procedures. The development of INDRA to date has focused on cell signaling, leaving metabolism, lipid biology, microRNA function, epigenetic regulation, etc. as future extensions. We are actively extending INDRA to include such processes by 1) updating processors to retrieve a wider range of information; 2) adding new *Statement* types; and 3) creating the necessary assembly procedures. Other areas of future development include automated retrieval of binding affinities and kinetic rates for parameter estimation. Encouragingly, the Path2Models software has shown that automated retrieval of kinetic parameters from databases is feasible for metabolic models (Büchel *et al*, 2013), and this approach may be adaptable to signaling pathways as well. Another direction for extension involves capturing observational in addition to mechanistic information. For instance, the experimental finding "*IRS-1 knockdown resulted in reduction of insulin stimulated Akt1 phosphorylation at Ser 473*." (Varma & Khandelwal, 2008) cannot be directly represented as a molecular mechanism. Literature and databases contain a wealth of such indirect, non-mechanistic information that could be used as biological constraints to infer or verify mechanistic models.

A system such as INDRA allowing biologists to "talk" to a machine about a biological pathway in natural language suggests the possibility that an improved machine could also "talk back" to the human user (Carvunis & Ideker, 2014). At its most basic level, such a system would allow humans and machines to jointly curate knowledge, thereby resolving ambiguities or errors in NLP or assembly. A more sophisticated machine would use its internal knowledge base to autonomously identify additional relevant reactions, inconsistencies in a user's input, or novel hypotheses arising from model simulation. A computer agent could interact with many human experts simultaneously, facilitating curation and modeling efforts by communities of biologists. We anticipate that such human-machine collaborative systems will be increasingly valuable in making sense of the large and complex datasets that characterize modern biology.

MATERIALS AND METHODS

Software and model availability

INDRA is available under the open-source BSD license. Code and documentation are available via http://indra.bio; the documentation is also included as part of the Supplementary Information. The TRIPS/DRUM system for extracting mechanisms from natural language is available at http://trips.ihmc.us/parser/cgi/drum. INDRA version 1.4.2 was used to obtain all results in the manuscript.

The POMI1.0 and MEMI1.0-1.2 models are provided as supplementary attachments in SBML, BNGL, Kappa and PySB formats, in addition to the natural language text files used to build them. The RAS pathway model and its extension are provided in SIF and Boolean network formats as supplementary attachments. Code used to generate these models is part of the INDRA repository and can be found in the *models* folder of https://github.com/sorgerlab/indra.

TRIPS Interface

The INDRA TRIPS *Interface* is invoked using the top-level function *process_text*. This function queries the TRIPS/DRUM web service via HTTP request, sending the natural language content as input and retrieving extracted events in the EKB-XML format. The *Interface* then creates an instance of the *TripsProcessor* class, which is then used to iteratively search the EKB-XML output, via XPath queries, for entries corresponding to INDRA *Statements*. Extracted *Statements* are stored in the *statements* property of the *TripsProcessor*, which is returned by the *Interface* to the calling function.

BioPAX/Pathway Commons Interface

INDRA's BioPAX *Interface* either queries the Pathway Commons web service or reads an offline BioPAX OWL file (Box 2). The *Interface* contains three functions that can be used to query the Pathway Commons database via the web service: 1) *process_pc_neighborhood,* which returns the reactions containing one or more query genes, 2) *process_pc_pathsbetween,* which returns reaction paths connecting the query genes, subject to a path length limit, and 3) *process_pc_pathsfromto,* which returns reaction paths from a source gene set to a target gene set, subject to a path length limit. The BioPAX *Interface* processes the resulting OWL files using PaxTools (Demir *et al*, 2013), yielding a BioPAX model as a Java object accessible in Python via the *pyjnius* Python-Java bridge (https://github.com/kivy/pyjnius). INDRA's BioPAX *Processor* then uses the BioPAX Patterns package

(Babur *et al*, 2014) to query the BioPAX object model for reaction patterns corresponding to INDRA *Statements*.

BEL/NDEx Interface

INDRA's BEL *Interface* either reads an offline BEL-RDF file or obtains BEL-RDF from the BEL Large Corpus via the Network Data Exchange (NDEx) web service (Pratt *et al*, 2015). Subnetworks of the BEL Large Corpus are obtained by calling the method *process_ndex_neighborhood,* which retrieves BEL Statements involving one or more query genes. The BEL *Processor* then uses the Python package *rdflib* to query the resulting RDF object for BEL Statements corresponding to INDRA *Statements* via the SPARQL Protocol and RDF Query Language (SPARQL; https://www.w3.org/TR/sparql11-overview).

Assembly of rule-based models

Assembly of rule-based models is performed by instances of the PySB *Assembler* class. Given a set of INDRA Statements and assembly policies as input, the *make_model* method of the PySB *Assembler* assembles models in two steps. First, information is collected about all molecular entities referenced by the set of *Statements*. This defines the activity types, post-translational modification sites, binding sites, and mutation sites for each Agent, which can then be used to generate the agent "signatures" for the rule-based model. In PySB, the molecular entities of the model are represented by a set of instances of the PySB *Monomer* class. Because assembly policies chosen by the user govern the nature of binding interactions (e.g., one-step vs. two-step modification), the binding sites and agent signatures must be generated in accordance with the chosen policies at this step. For policies involving explicit binding between proteins (e.g., the *two-step* policy for post-translational modifications), each PySB *Monomer* is given a unique binding site for each interacting partner. The second step is the generation of reaction rules corresponding to each of the input Statements. The PySB *Assembler* iteratively processes each *Statement*, calling the assembly function specific to the Statement type and chosen policy. Depending on the *Statement* type and policy, one or more PySB rules may be generated and added to the PySB model. The PySB model returned by the *make_model* function can then be converted into other formats (Kappa, BNG, SBML, Matlab, etc.) depending on the type of simulation or analysis to be performed (Lopez *et al*, 2013). Importantly, the PySB *Assembler* adds annotations to the generated PySB model that link molecular entities referenced in the model to their identities in reference ontologies (e.g., HGNC and UniProt). These annotations are in turn propagated into SBML and other model formats by existing PySB model export routines.

Models of p53 activation in response to single- and double strand break DNA damage

The text defining each model was submitted to the TRIPS web service for processing via INDRA's TRIPS *Interface*. The TRIPS system returned Extraction Knowledge Base graphs (Box 1 and Supplementary Information section 2.2) from which INDRA's TRIPS *Processor* extracted INDRA *Statements*. These *Statements* were then assembled using INDRA's PySB *Assembler* into a rule-based model. The default "one-step" assembly policy was used, which generates rules in which the subject of an activation, inhibition, and modification changes the state of the object without binding.

The 8 sentences constituting the SSB damage response model (Figure 5B) resulted in 8 INDRA *Statements* (each of type Activation or Inhibition). For example, the sentence "Active p53 activates Mdm2" was represented as an Activation *Statement* with an additional condition on the *Agent* representing p53, requiring that it be active. During INDRA *Statement* construction, names of genes are standardized to their HGNC gene symbol (Eyre *et al*, 2006), thus, the *Agent* representing "Mdm2" is renamed "MDM2", and the *Agent* representing "p53" is renamed "TP53". Default initial conditions (10,000 molecules, based on a default concentration of 10^{-8} Molar in a typical HeLa cell volume of 1.6 x 10^{-12} L) generated by the PySB *Assembler* were used for each protein in its inactive state and simulations were started with an initial 1 active ATR molecule to initiate the activation pathway. The forward rates for activation and inhibition rules were set to 10^{-7} $molec^{-1}s^{-1}$ (using a conversion rate of 10^5 $M^{-1}s^{-1}$ in a typical HeLa cell volume, as above). The forward rate of the rules corresponding to ATR auto-activation and p53 inactivation by Wip1 were modified to be 5 x 10^{-7} $molec^{-1}s^{-1}$, that is, faster than the forward rate of other rules (a summary of all rules and rates is given in the Supplementary Information section 2.2). PySB's reaction network generation and simulation functions were then used to instantiate the model as a set of 8 ordinary differential equations. The model was simulated using the *scipy* package's built-in *vode* solver for up to 20 hours of model time while tracking the amount of active p53, which was then plotted (Figure 5B). Natural language processing for this model took 10 seconds (here and in the following this includes network traffic time to and from the web service); the assembly and simulation of the model took less than 1 second.

The method for constructing the simple DSB response model (Figure 5C) with ATM was analogous to the SSB model. The same initial amounts and forward rate constants were used as in the previous model, except in this case an initial condition of 1 active ATM molecule was used, and the inactivation of ATM by Wip1 was given a forward rate of 10^{-5} $molec^{-1}s^{-1}$. For this model, the 9 natural language sentences were captured in 9 INDRA *Statements* and generated into a model of 9 rules and finally 9 ODEs. The model was again simulated up to 20 hours while observing the active form of p53.

Similar to the SSB response model, natural language processing for this model took around 10 seconds, with assembly and simulation taking less than 1 second.

The POMI1.0 model (Figure 5E) extends the basic DSB response model by specifying the activation/inhibition processes in more mechanistic detail. The model is described in 10 sentences yielding 12 INDRA *Statements* and a model containing 11 PySB rules and 12 ODEs (via the PySB *Assembler* using the “one-step” policy). The same rate constants were used as in the simple DSB response model; additionally, the degradation rate of Mdm2 was set to 8×10^{-2} s^{-1} and the rate of synthesis of Mdm2 by p53 to 2×10^{-2} $molec^{-1}s^{-1}$ (a full list of rules and associated rate constants is given in the Supplementary Information section 2.2). Natural language processing for this model took 14 seconds; assembly and simulation took less than 1 second.

Models of response to BRAF inhibition

The sentences for the MEMI1.0, 1.1 and 1.2 models were processed with the TRIPS web service via INDRA’s TRIPS *Interface*. Natural language processing took 37 seconds for MEMI1.0, 60 seconds for MEMI1.1, and 75 seconds for MEMI1.2. The resulting INDRA *Statements* were then assembled using INDRA’s PySB *Assembler* module into a rule-based model using the “two-step” policy for assembling post-translational modifications. Kinetic rate constants were set manually and the initial amounts of each protein were set to correspond in their order of magnitude to typical absolute copy numbers measured across a panel of cancer cell lines in Table S5 of (Shi *et al*, 2016). A summary of the kinetic rates and initial amounts is given in Supplementary Tables 4-6. Each model was instantiated as a system of ordinary differential equations and simulated using the *scipy* Python package’s built-in *vode* solver. Each model was started from an initial condition with all proteins in an inactive, unmodified and unbound state. The models were run to steady state and the values of GTP-bound RAS (active RAS) and phosphorylated ERK were saved. Another simulation was then started from the steady state values with vemurafenib added and the time courses of active RAS and phosphorylated ERK were normalized against their unperturbed steady state values and plotted.

Extensible and executable RAS pathway map

The pathway map was created by processing 47 sentences with TRIPS (see Supplementary Information section 2.4) to generate 141 INDRA *Statements*. Reading and extraction of *Statements* took a total of 160 seconds. The *Statements* were then assembled using INDRA’s *Graph Assembler*, which produced a network that was laid out using Graphviz (Ellson *et al*, 2002) as shown in Figure 7A. The same set of *Statements* was then assembled using the INDRA SIF *Assembler* which produced a list of positive and negative interactions between genes that can be interpreted by network visualization software (Shannon *et al*, 2003) and Boolean network simulation tools. The logical functions for each node were generated by combining the state of parent nodes such that the presence of any activating input in an *on* state and the absence of any inhibitory inputs in an *on* state resulted in the node's value taking an *on* state at the next time step (logical rules are given in Supplementary Information section 2.4). Boolean network simulations were performed using the *boolean2* package (Albert *et al*, 2008). First, 100 independent traces were simulated using asynchronous updates on the nodes (which results in stochastic behavior) and the average of the value of each node (with 0 corresponding to the low and 1 to the high state of each node) was taken across all simulations to produce the time course plots in Figure 7D.

ACKNOWLEDGEMENTS

This work was funded by ARO grant W911NF-14-1-0397 to PKS under the DARPA Big Mechanism program and by NIGMS grant P50GM107618 to PKS.

AUTHOR CONTRIBUTIONS

BMG and JAB designed and implemented INDRA. BMG, JAB, KS, JM, LG and PKS conceived the overall approach. BMG, JAB, KS, and PKS wrote the paper.

CONFLICT OF INTEREST

The authors declare that they have no conflict of interest.

REFERENCES

Agrawal A, Yang J, Murphy RF & Agrawal DK (2006) Regulation of the p14ARF-Mdm2-p53 pathway: An overview in breast cancer. *Exp. Mol. Pathol.* **81:** 115–122

Albeck JG, Burke JM, Aldridge BB, Zhang M, Lauffenburger DA & Sorger PK (2008) Quantitative Analysis of Pathways Controlling Extrinsic Apoptosis in Single Cells. *Mol. Cell* **30:** 11–25

Albert I, Thakar J, Li S, Zhang R & Albert R (2008) Boolean network simulations for life scientists. *Source Code Biol. Med.* **3:** 16

Allen J, de Beaumont W, Galescu L & Teng CM (2015) Complex Event Extraction using DRUM. In *ACL-IJCNLP* pp 1–11. Beijing, China

Allen J, Ferguson G, Blaylock N, Byron D, Chambers N, Dzikovska M, Galescu L & Swift M (2006) Chester: Towards a personal medication advisor. *J. Biomed. Inform.* **39:** 500–513

Allen JF (2003) Natural language processing. *Encycl. Comput. Sci.***:** 1218–1222

Anafi M, Kiefer F, Gish GD, Mbamalu G, Iscove NN & Pawson T (1997) SH2/SH3 adaptor proteins can link tyrosine kinases to a Ste20-related protein kinase, HPK1. *J. Biol. Chem.* **272:** 27804–27811

Antonyak MA, Moscatello DK & Wong AJ (1998) Constitutive activation of c-Jun N-terminal kinase by a mutant epidermal growth factor receptor. *J Biol Chem* **273:** 2817–2822

Babur Ö, Aksoy BA, Rodchenkov I, Sümer SO, Sander C & Demir E (2014) Pattern search in BioPAX models. *Bioinformatics* **30:** 139–140

Bakkenist CJ & Kastan MB (2003) DNA damage activates ATM through intermolecular autophosphorylation and dimer dissociation. *Nature* **421:** 499–506

Batchelor E, Loewer A, Mock C & Lahav G (2011) Stimulus-dependent dynamics of p53 in single cells. *Mol. Syst. Biol.* **7:** 488

Birtwistle MR, Hatakeyama M, Yumoto N, Ogunnaike B a, Hoek JB & Kholodenko BN (2007) Ligand-dependent responses of the ErbB signaling network: experimental and modeling analyses. *Mol. Syst. Biol.* **3:** 144

Büchel F, Rodriguez N, Swainston N, Wrzodek C, Czauderna T, Keller R, Mittag F, Schubert M, Glont M, Golebiewski M, van Iersel M, Keating S, Rall M, Wybrow M, Hermjakob H, Hucka M, Kell DB, Müller W, Mendes P, Zell A, et al (2013) Path2Models: large-scale generation of computational models from biochemical pathway maps. *BMC Syst. Biol.* **7:** 116

Carvunis AR & Ideker T (2014) Siri of the cell: What biology could learn from the iPhone. *Cell* **157:** 534–538

Catlett NL, Bargnesi AJ, Ungerer S, Seagaran T, Ladd W, Elliston KO & Pratt D (2013) Reverse causal reasoning: applying qualitative causal knowledge to the interpretation of high-throughput data. *BMC Bioinformatics* **14:** 340

Cerami EG, Gross BE, Demir E, Rodchenkov I, Babur Ö, Anwar N, Schultz N, Bader GD & Sander C (2011) Pathway Commons, a web resource for biological pathway data. *Nucleic Acids Res.* **39:** 685–690

Chambers N, Allen J, Galescu L & Jung H (2005) A dialogue-based approach to multi-robot team control. *Multi-Robot Syst. From***:** 1–7

Chen KC, Calzone L, Csikasz-Nagy A, Cross FR, Novak B & Tyson JJ (2004) Integrative analysis of cell cycle control in budding yeast. *Mol. Biol. Cell* **15:** 3841–62

Chen WW, Schoeberl B, Jasper PJ, Niepel M, Nielsen UB, Lauffenburger DA & Sorger PK (2009) Input-output behavior of ErbB signaling pathways as revealed by a mass action model trained against dynamic data. *Mol. Syst. Biol.* **5:** 239

Choi D, Na W, Kabir M, Yi E, Kwon S, Yeom J, Ahn JW, Choi HH, Lee Y, Seo K, Shin M, Park SH, Yoo H, Isono K ichi, Koseki H, Kim ST, Lee C, Kwon Y & Choi C (2013) WIP1, a Homeostatic Regulator of the DNA Damage Response, Is Targeted by HIPK2 for Phosphorylation and Degradation. *Mol. Cell* **51:** 374–385

Corbalan-Garcia S, Yang SS, Degenhardt KR & Bar-Sagi D (1996) Identification of the mitogen-activated protein kinase phosphorylation sites on human Sos1 that regulate interaction with Grb2. *Mol. Cell. Biol.* **16:** 5674–5682

Danos V, Feret J, Fontana W, Harmer R & Krivine J (2007a) Rule-based modelling of cellular signalling. In *CONCUR 2007–Concurrency Theory* pp 17–41. Springer

Danos V, Feret J, Fontana W, Harmer R & Krivine J (2009) Rule-based modelling and model perturbation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* pp 116–137.

Danos V, Feret J, Fontana W & Krivine J (2007b) Scalable simulation of cellular signaling networks. *Program. Lang. Syst.*: 139–157

Demir E, Babur Ö, Rodchenkov I, Aksoy BA, Fukuda KI, Gross B, Sümer OS, Bader GD & Sander C (2013) Using Biological Pathway Data with Paxtools. *PLoS Comput. Biol.* **9:**

Demir E, Cary MP, Paley S, Fukuda K, Lemer C, Vastrik I, Wu G, D'Eustachio P, Schaefer C, Luciano J, Schacherer F, Martinez-Flores I, Hu Z, Jimenez-Jacinto V, Joshi-Tope G, Kandasamy K, Lopez-Fuentes AC, Mi H, Pichler E, Rodchenkov I, et al (2010) BioPAX -- A Community Standard for Pathway Data Sharing. *Nat Biotechnol* **28:**

Dhillon AS, Hagan S, Rath O & Kolch W (2007) MAP kinase signalling pathways in cancer. *Oncogene* **26:** 3279–3290

Ellson J, Gansner E, Koutsofios L, North SC & Woodhull G (2002) Graphviz – Open Source Graph Drawing Tools. *Graph Draw.*: 483–484

Eydgahi H, Chen WW, Muhlich JL, Vitkup D, Tsitsiklis JN & Sorger PK (2013) Properties of cell death models calibrated and compared using Bayesian approaches. *Mol. Syst. Biol.* **9:** 644

Eyre TA, Ducluzeau F, Sneddon TP, Povey S, Bruford EA & Lush MJ (2006) The HUGO Gene Nomenclature Database, 2006 updates. *Nucleic Acids Res.* **34:** D319–D321

Faeder JR, Blinov ML, Goldstein B & Hlavacek WS (2005) Combinatorial complexity and dynamical restriction of network flows in signal transduction. *Syst. Biol. (Stevenage).* **2:** 5–15

Faeder JR, Blinov ML & Hlavacek WS (2009) Rule-Based Modeling of Biochemical Systems with BioNetGen. In *Methods in Molecular Biology, Systems Biology* pp 83–89.

Feret J, Danos V, Krivine J, Harmer R & Fontana W (2009) Internal coarse-graining of molecular systems. *Proc. Natl. Acad. Sci. U. S. A.* **106:** 6453–6458

Ferguson G & Allen J (1998) TRIPS: An integrated intelligent problem-solving assistant. *Aaai/Iaai***:** 567–572

Fey D, Halasz M, Dreidax D, Kennedy SP, Hastings JF, Rauch N, Munoz AG, Pilkington R, Fischer M, Westermann F, Kolch W, Kholodenko BN & Croucher DR (2015) Signaling pathway models as biomarkers : Patient-specific simulations of JNK activity predict the survival of neuroblastoma patients. *Science (80-.).* **8:** RA130

Fluck J & Hofmann-Apitius M (2014) Text mining for systems biology. *Drug Discov. Today* **19:** 140–144

Gunawardena J (2014a) Models in biology: "accurate descriptions of our pathetic thinking". *BMC Biol.* **12:** 29

Gunawardena J (2014b) Time-scale separation - Michaelis and Menten's old idea, still bearing fruit. *FEBS J.* **281:** 473–488

Harmer R, Danos V, Feret J, Krivine J & Fontana W (2010) Intrinsic information carriers in combinatorial dynamical systems. *Chaos* **20:**

Heinrich R, Neel BG & Rapoport TA (2002) Mathematical models of protein kinase signal transduction. *Mol Cell* **9:** 957–970

Joseph EW, Pratilas CA, Poulikakos PI, Tadi M, Wang W, Taylor BS, Halilovic E, Persaud Y, Xing F, Viale A, Tsai J, Chapman PB, Bollag G, Solit DB & Rosen N (2010) The RAF inhibitor PLX4032 inhibits ERK signaling and tumor cell proliferation in a V600E BRAF-selective manner. *Proc Natl Acad Sci U S A* **107:** 14903–14908

Kahramanoğullari O, Cardelli L & Caron E (2009) An Intuitive Automated Modelling Interface for Systems Biology. *Electron. Proc. Theor. Comput. Sci.* **9:** 73–86

Karr JR, Sanghvi JC, MacKlin DN, Gutschow M V., Jacobs JM, Bolival B, Assad-Garcia N, Glass JI & Covert MW (2012) A whole-cell computational model predicts phenotype from genotype. *Cell* **150:** 389–401

Kholodenko BN (2015) Drug Resistance Resulting from Kinase Dimerization Is Rationalized by Thermodynamic Factors Describing Allosteric Inhibitor Effects. *Cell Rep.* **12:** 1939–1949

Kolpakov F, Puzanov M & Koshukov A (2006) BioUML: visual modeling, automated code generation and simulation of biological systems. In *Proceedings of The Fifth International Conference on Bioinformatics of Genome Regulation and Structure* pp 281–284.

Krallinger M, Leitner F, Vazquez M, Salgado D, Marcelle C, Tyers M, Valencia A & Chatr-aryamontri A (2012) How to link ontologies and protein--protein interactions to literature: text-mining approaches and the BioCreative experience. *Database* **2012:** bas017

Larkin J, Ascierto PA, Dréno B, Atkinson V, Liszkay G, Maio M, Mandalà M, Demidov L, Stroyakovskiy D, Thomas L, de la Cruz-Merino L, Dutriaux C, Garbe C, Sovak MA, Chang I, Choong N, Hack SP, McArthur GA & Ribas A (2014) Combined vemurafenib and cobimetinib in BRAF-mutated melanoma. *N. Engl. J. Med.* **371:** 1867–76

Lavoie H, Thevakumaran N, Gavory G, Li JJ, Padeganeh A, Guiral S, Duchaine J, Mao DY, Bouvier M, Sicheri F & Therrien M (2013) Inhibitors that stabilize a closed RAF kinase domain conformation induce dimerization. *Nat Chem Biol*

Lindner AU, Concannon CG, Boukes GJ, Cannon MD, Llambi F, Ryan D, Boland K, Kehoe J, McNamara DA, Murray F, Kay EW, Hector S, Green DR, Huber HJ & Prehn JHM (2013) Systems analysis of BCL2 protein family interactions establishes a model to predict responses to chemotherapy. *Cancer Res.* **73:** 519–528

Lito P, Pratilas CA, Joseph EW, Tadi M, Halilovic E, Zubrowski M, Huang A, Wong WL, Callahan MK, Merghoub T, Wolchok JD, de Stanchina E, Chandarlapaty S, Poulikakos PI, Fagin JA & Rosen N (2012) Relief of Profound Feedback Inhibition of Mitogenic Signaling by RAF Inhibitors Attenuates Their Activity in BRAFV600E Melanomas. *Cancer Cell* **22:** 668–682

Lito P, Rosen N & Solit DB (2013) Tumor adaptation and resistance to RAF inhibitors. *Nat. Med.* **19:** 1401–9

Liu S, Shiotani B, Lahiri M, Maréchal A, Tse A, Leung CCY, Glover JNM, Yang XH & Zou L (2011) ATR Autophosphorylation as a Molecular Switch for Checkpoint Activation. *Mol. Cell* **43:** 192–202

Loew LM & Schaff JC (2001) The Virtual Cell: A software environment for computational cell biology. *Trends Biotechnol.* **19:** 401–406

Lopez CF, Muhlich JL, Bachman JA & Sorger PK (2013) Programming biological models in Python using PySB. *Mol. Syst. Biol.* **9:** 646

Mallavarapu A, Thomson M, Ullian B & Gunawardena J (2009) Programming with models: modularity and abstraction provide powerful capabilities for systems biology. *J. R. Soc. Interface* **6:** 257–270

Manshadi MH, Allen J & Swift M (2008) Toward a universal underspecified semantic representation. In *13th Conference on Formal Grammar (FG 2008), Hamburg, Germany*

Mendes P & Kell DB (1998) Non-linear optimization of biochemical pathways: Applications to metabolic engineering and parameter estimation. *Bioinformatics* **14:** 869–883

Moles CG, Mendes P & Banga JR (2003) Parameter estimation in biochemical pathways: A comparison of global optimization methods. *Genome Res.* **13:** 2467–2474

Montagut C, Sharma S V., Shioda T, McDermott U, Ulman M, Ulkus LE, Dias-Santagata D, Stubbs H, Lee DY, Singh A, Drew L, Haber DA & Settleman J (2008) Elevated CRAF as a potential mechanism of acquired resistance to BRAF inhibition in melanoma. *Cancer Res.* **68:** 4853–4861

Novák B & Tyson JJ (2008) Design principles of biochemical oscillators. *Nat. Rev. Mol. Cell Biol.* **9:** 981–91

Le Novère N, Finney A, Hucka M, Bhalla US, Campagne F, Collado-Vides J, Crampin EJ, Halstead M, Klipp E, Mendes P, Nielsen P, Sauro H, Shapiro B, Snoep JL, Spence HD & Wanner BL (2005) MIRIAM, Minimum information requested in the annotation of biochemical models. *Nat Biotechnol* **23:**

Le Novère N, Hucka M, Mi H, Moodie S, Schreiber F, Sorokin A, Demir E, Wegner K, Aladjem MI, Wimalaratne SM, Bergman FT, Gauges R, Ghazal P, Kawaji H, Li L, Matsuoka Y, Villéger A, Boyd SE, Calzone L, Courtot M, et al (2009) The Systems Biology Graphical Notation. *Nat. Biotechnol.* **27:** 735–741

Oliphant TE (2007) SciPy: Open source scientific tools for Python. *Comput. Sci. Eng.* **9:** 10–20

Poulikakos PI, Zhang C, Bollag G, Shokat KM & Rosen N (2010) RAF inhibitors transactivate RAF dimers and ERK signalling in cells with wild-type BRAF. *Nature* **464:** 427–430

Pratt D, Chen J, Welker D, Rivas R, Pillich R, Rynkov V, Ono K, Miello C, Hicks L, Szalma S, Stojmirovic A, Dobrin R, Braxenthaler M, Kuentzer J, Demchak B & Ideker T (2015) NDEx, the Network Data Exchange. *Cell Syst.* **1:** 302–305

Purvis JE, Karhohs KW, Mock C, Batchelor E, Loewer A & Lahav G (2012) p53 Dynamics Control Cell Fate. *Science (80-.).* **336:** 1440–1444

Purvis JE & Lahav G (2013) Encoding and decoding cellular information through signaling dynamics. *Cell* **152:** 945–956

Rehm M, Düßmann H, Jänicke RU, Tavaré JM, Kögel D & Prehn JHM (2002) Single-cell fluorescence resonance energy transfer analysis demonstrates that caspase activation during apoptosis is a rapid process: Role of caspase-3. *J. Biol. Chem.* **277:** 24506–24514

Ruebenacker O, Moraru II, Schaff JC & Blinov ML (2009) Integrating BioPAX pathway knowledge with SBML models. *IET Syst. Biol.* **3:** 317–328

Salazar C & Höfer T (2006) Kinetic models of phosphorylation cycles: A systematic approach using the rapid-equilibrium approximation for protein-protein interactions. In *BioSystems* pp 195–206.

Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski B & Ideker T (2003) Cytoscape: A software Environment for integrated models of biomolecular interaction networks. *Genome Res.* **13:** 2498–2504

Shi H, Moriceau G, Kong X, Lee M-K, Lee H, Koya RC, Ng C, Chodon T, Scolyer RA, Dahlman KB, Sosman JA, Kefford RF, Long G V, Nelson SF, Ribas A & Lo RS (2012a) Melanoma whole-exome sequencing identifies (V600E)B-RAF amplification-mediated acquired B-RAF inhibitor resistance. *Nat. Commun.* **3:** 724

Shi H, Moriceau G, Kong X, Lee M-K, Lee H, Koya RC, Ng C, Chodon T, Scolyer RA, Dahlman KB, Sosman JA, Kefford RF, Long G V, Nelson SF, Ribas A & Lo RS (2012b) Melanoma whole-exome sequencing identifies (V600E)B-RAF amplification-mediated acquired B-RAF inhibitor resistance. *Nat. Commun.* **3:** 724 Available at: http://www.nature.com/authors/editorial_policies/license.html#terms%5Cnhttp://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3530385&tool=pmcentrez&rendertype=abstract

Shi T, Niepel M, McDermott JE, Gao Y, Nicora CD, Chrisler WB, Markillie LM, Petyuk VA, Smith RD, Rodland KD, Sorger PK, Qian W-J & Wiley HS (2016) Conservation of protein abundance patterns reveals the regulatory architecture of the EGFR-MAPK pathway. *Sci. Signal.* **9:** 1–14

Smith LP, Bergmann FT, Chandran D & Sauro HM (2009) Antimony: A modular model definition language. *Bioinformatics* **25:** 2452–2454

Sneddon MW, Faeder JR & Emonet T (2011) Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. *Nat. Methods* **8:** 177–183

Stefan MI, Bartol TM, Sejnowski TJ & Kennedy MB (2014) Multi-state Modeling of Biomolecules. *PLoS Comput. Biol.* **10:**

Stephen AG, Esposito D, Bagni RK & McCormick F (2014) Dragging ras back in the ring. *Cancer Cell* **25:** 272–281

Stites EC, Trampont PC, Ma Z & Ravichandran KS (2007) Network analysis of oncogenic Ras activation in cancer. *Science (80-.).* **318:** 463–467

Thomas BR, Chylek LA, Colvin J, Sirimulla S, Clayton AHA, Hlavacek WS & Posner RG (2015) BioNetFit: A fitting tool compatible with BioNetGen, NFsim and distributed computing environments. *Bioinformatics* **32:** 798–800

Tiger CF, Krause F, Cedersund G, Palmer R, Klipp E, Hohmann S, Kitano H & Krantz M (2012) A framework for mapping, visualisation and automatic model creation of signal-transduction networks. *Mol Syst Biol* **8:** 578

Turei D, Korcsmaros T & Saez-Rodriguez J (2016a) OmniPath: guidelines and gateway for literature-curated signaling pathway resources. *Nat Meth* **13:** 966–967

Turei D, Korcsmaros T & Saez-Rodriguez J (2016b) OmniPath: guidelines and gateway for literature-curated signaling pathway resources. *Nat Meth* **13:** 966–967 Available at: http://dx.doi.org/10.1038/nmeth.4077

Valenzuela-Escarcega MA, Gus H-P, Thomas H & Surdeanu M (2015) A Domain-independent Rule-based Framework for Event Extraction. *Proc. 53rd Annu. Meet. Assoc. Comput. Linguist. 7th Int. Jt. Conf. Nat. Lang. Process. (Volume 1 Long Pap.***:** 127–132

Varma S & Khandelwal RL (2008) Overexpression of Akt1 upregulates glycogen synthase activity and phosphorylation of mTOR in IRS-1 knockdown HepG2 cells. *J. Cell. Biochem.* **103:** 1424–1437

Wagner EF & Nebreda AR (2009) Signal integration by JNK and p38 MAPK pathways in cancer development. *Nat Rev Cancer* **9:** 537–549

Wasik S, Prejzendanc T & Blazewicz J (2013) ModeLang: A new approach for experts-friendly viral infections modeling. *Comput. Math. Methods Med.* **2013:**

Wrzodek C, Büchel F, Ruff M, Dräger A & Zell A (2013) Precise generation of systems biology models from KEGG pathways. *BMC Syst. Biol.* **7:** 15 Available at: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3623889&tool=pmcentrez&rendertype=abstract

Yao Z, Torres NM, Tao A, Luo L, Abdel-wahab OI, Solit D, Poulikakos P & Rosen N (2015) BRAF mutants evade ERK dependent feedback by different mechanisms that determine their sensitivity to pharmacologic inhibition. *Cancer Cell***:** 370–383

FIGURE LEGENDS

Figure 1. Building a model from natural language with INDRA.

(A) The architecture of INDRA consists of three layers of modules (1-3). In layer (1), Interfaces collect mechanisms from natural language processing systems (e.g. TRIPS *Interface*) and pathway databases (e.g. Pathway Commons *Interface*) and Processors (e.g. TRIPS *Processor*, BioPAX *Processor*) extract INDRA *Statements* from their outputs. Statements, the internal representation in INDRA constitute layer (2). In layer (3), INDRA *Statements* are assembled into various model formats by *Assembler* modules (e.g. PySB *Assembler*, Graph *Assembler*).

(B) A Python script is used to assemble and simulate a model from the *text* "MEK1 phosphorylates ERK2 at threonine 185 and tyrosine 187". The *process_text* method of INDRA's TRIPS *Processor* is called to send the text to the TRIPS NLP system (1) and then process the output of TRIPS to construct INDRA *Statements* (2). Then, a PySB *Assembler* is constructed, the *Statements* are added to it, and an executable model is assembled using the PySB *Assembler*'s *make_model* method with a "two-step" policy (3). Finally, the model is simulated for 300 seconds using PySB's *odesolve* function.

(C) User input, INDRA modules and external tools form a sequence of events to turn a natural language sentence into a model and simulation. The natural language description from the user is passed to INDRA's TRIPS *Interface*, which sends the text to TRIPS (1). The TRIPS system processes the text and creates and Extraction Knowledge Base graph (Results column; yellow box). INDRA receives the results from TRIPS and constructs two INDRA *Statements* from it, one for each phosphorylation event (Results column), which are returned to the user (2). The user then instantiates a PySB *Assembler* and instructs it to assemble an executable model (3) from the given INDRA *Statements* (a schematic biochemical reaction network shown in Results column). Finally, the user calls an ODE solver via PySB's *odesolve* function to simulate the model for 300 seconds (simulation output shown in Results column).

Figure 2. INDRA Statements represent molecular agents and biochemical mechanisms.

(A) The mechanism "MAP2K1 that is phosphorylated at S218 and S222 phosphorylates MAPK1 on T185" is represented in INDRA as a Phosphorylation *Statement* with an enzyme *Agent* (MAP2K1), a substrate *Agent* (MAPK1), a residue (Threonine), and a position (185) argument. The state of the MAP2K1 *Agent* is expanded in panel (B). A *Statement* can have one or more *Evidences* associated with it, with an example expanded in panel (C).

(B) The *Agent* representing "MAP2K1 that is phosphorylated at S218 and S222" has two modification conditions: serine-phosphorylation at 218 and serine-phosphorylation at 222. The grounding to the UniProt and HGNC databases associated with the *Agent* is also shown.

(C) An *Evidence* object is shown which is associated with an INDRA *Statement* obtained from the BEL Large Corpus (see Box 2) as the source. The *Evidence* object represents the evidence text for the entry ("c-Raf activates MEK1 by phosphorylating at serine residues 218 and 222"), the citation associated with the entry (PubMed identifier 8621729), the original BEL statement (shown under Source ID) and any annotations that are available, including the organism (in this example, 9606, which is the identifier for *Homo sapiens*). In some cases, epistemic information is known about the *Statement*, such as whether it is an assertion or a hypothesis, and the *Evidence* object has a corresponding field to carry this information.

Figure 3. INDRA Statements constructed from TRIPS NLP extractions, BioPAX and BEL. An identical INDRA *Statement* is constructed from three knowledge sources. A corresponding fragment of each source format (representing the phosphorylated state of MAP2K1 on S222) is highlighted in blue.

Top left: A TRIPS EKB (see Box 1) graph is shown for the sentence "MAP2K1 that is phosphorylated on S218 and S222 phosphorylates MAPK1 at T185". The main phosphorylation *event* has *agent*, *affected* and *site* arguments, which each refer to a *term*. The *agent term* resolves to a *gene* with name MAP2K1 and database references to UniProt and HGNC. The MAP2K1 *term* also refers to an additional *event* in which it is *affected* (yellow background). This additional event represents the phosphorylated state at two molecular sites: serine 218 and serine 222. The *affected* Term associated with the main phosphorylation event is MAPK1 with its associated UniProt and HGNC references. Finally, the site argument of the main event is a *molecular-site* resolving to threonine 185.

Middle left: A BioPAX *Biochemical Reaction* is shown with unmodified MAPK1 on the left hand side and MAPK1 with a *Sequence Modification Feature* of phosphorylation at threonine 185 on the right hand side. Both the left and the right hand sides use the same *Cross Reference* to a UniProt identifier. A *Catalysis* is associated with the *Biochemical Reaction* with MAP2K1 as the controller. MAP2K1 has two *Sequence Modification Features*: phosphorylation at serines 218 and 222. MAP2K1 also refers to a UniProt identifier via a *Cross Reference*. Two alternative visual representations of the same BioPAX *Reaction* are given in Supplementary Figure S4.

Bottom left: A graphical representation of a BEL statement is shown in which the *subject* is the *Kinase Activity* of the *Protein Abundance* of the modified MAP2K1 (with phosphorylations at serines 218 and 222). The *object* of the statement is the *Protein Abundance* of modified MAPK1 (phosphorylation at threonine 185) with the predicate being *Directly Increases*. Below the graphical representation, the statement is also given in *BEL script* format.

Right: All example mechanisms from the three knowledge sources are constructed as the same INDRA *Phosphorylation Statement* with MAP2K1 as the enzyme (subject to modification conditions) and MAPK1 and the substrate. The *Evidence* associated with the INDRA *Statement* (not shown) constructed would be different for each knowledge source.

Figure 4. INDRA Statements are assembled into biochemical rules via assembly policies

The flow from representation and model content to implementation is governed by assembly policies and biochemical rule templates (top). A phosphorylation INDRA *Statement* with enzyme (MAP2K1) and substrate (MAPK1) can be assembled using several policies including one-step (top center), two-step (middle center) and ATP-dependent (bottom center). Each policy corresponds to a template for a generic enzyme (E) and a substrate (S). The one-step policy assumes that the enzyme catalyzes the phosphorylation of the substrate in a single step such that that the transient enzyme-substrate complex is not modeled. This is represented as a single rule (Rule 1; red box) instantiated as a PySB rule (top right). The two-step policy assumes the reversible formation of an enzyme-substrate complex and an irreversible catalysis and product release step corresponding to two overlapping rules (Rules 1-2; red boxes). The ATP-dependent policy assumes a template in which the enzyme has to bind both the substrate and ATP but can bind them in an arbitrary order. This corresponds to two rules: one for ATP binding and one for substrate binding. A third rule describes the release of the phosphorylated substrate from the enzyme-substrate complex (Rules 1-3; red boxes).

Figure 5. Modeling patterns of p53 activation dynamics from natural language

(A) Patterns of p53 activation dynamics upon double strand break DNA damage (left) and single strand break DNA damage (right), adapted from (Purvis & Lahav, 2013). Edges with yellow numbers correspond to the original diagram in (Purvis & Lahav, 2013), pink and green numbers correspond to mechanisms added subsequently, as described in the text.

(B) Natural language descriptions of the mechanisms involved in double strand break DNA damage (DSB) response corresponding to the diagram on the left hand side of (A) and dynamical simulation of p53 activity from the corresponding INDRA-assembled model (below).

(C) Natural language descriptions of the mechanisms involved in single strand break DNA damage (SSB) response corresponding to the diagram on the right hand side of (A) and dynamical simulation of p53 activity from the corresponding INDRA-assembled model (below).

(D) For the base sentence "Wip1 inactivates ATM", variants in the names of entities are shown below with four examples that produce the intended result (green sidebar) and one example that does not (red sidebar). To the right, eleven linguistic variants of the sentence are shown with eight producing the intended result (green sidebar) and three that do not, including one with a grammatical error and one with a spelling error (red sidebar).

(E) The POMI1.0 model, a variant of the double strand break response model with a mechanistically more detailed description of the system in the left hand side diagram in (A) and the model in (B). The model assembled with INDRA produces oscillations in p53 activity over time when simulated (bottom).

Figure 6. INDRA-built models of vemurafenib resistance in response to growth factor signals.
(A) Simplified schematic representation of the observed ERK phosphorylation phenomena in BRAF-V600E mutants that are hypothesized to be the basis of adaptive resistance. In untreated BRAF-V600E cells (left) mutant BRAF is constitutively active independently of RAS and leads to higher ERK phosphorylation levels (thick green edge) and stronger negative feedback to SOS (thick red edge). Upon vemurafenib treatment, in the short term (center) ERK phosphorylation is decreased due to BRAF V600E inhibition (thin green edge). Over time, resistance develops (right); the ERK-SOS feedback loop becomes weaker (thin red edge) and increased RAS activity induces RAF dimerization, leading to a rebound in ERK phosphorylation (thick green edge).

(B) MEMI1.0 is described in 14 sentences which are assembled into 28 PySB rules and generated into a system of 99 ordinary differential equations. Simulation of phosphorylated ERK (blue) and active RAS (green) are shown relative to their respective values at time 0, when vemurafenib is added. The model simulation shows that upon vemurafenib addition, the amount of phosphorylated ERK is quickly reduced and stays at a low level, while the amount of active RAS is unchanged.

(C) In MEMI1.1, by extending three existing sentences (4, 5, 14) and adding two new ones (15, 16) (changes shown in orange), the ERK-SOS negative feedback is modeled and assembled into 34 rules and 275 ODEs. The model simulation (right) reproduces RAS reactivation (green) upon vemurafenib treatment, however, the experimentally observed rise in ERK phosphorylation (blue) is not reproduced.

(D) MEMI1.2 extends MEMI1.1 by adding a sentence (17) and replacing an existing sentence with two new sentences (8A and 8B) (changes shown in green). INDRA produces a model consisting of 37 rules and 353 ODEs. Model simulations are able to reproduce the expected rise in RAS activation (green) and the increased phosphorylation of ERK (blue).

Figure 7. An INDRA-assembled extensible and executable pathway map of RAS signaling.
(A) Positive and negative activations as well as complex formation between proteins is written in natural language (left) to describe simplified interactions in the RAS pathway (for full text see Supplementary Information section 2.4). The INDRA-assembled graph is shown on the right showing activations (black), inhibitions (red) and binding (blue).

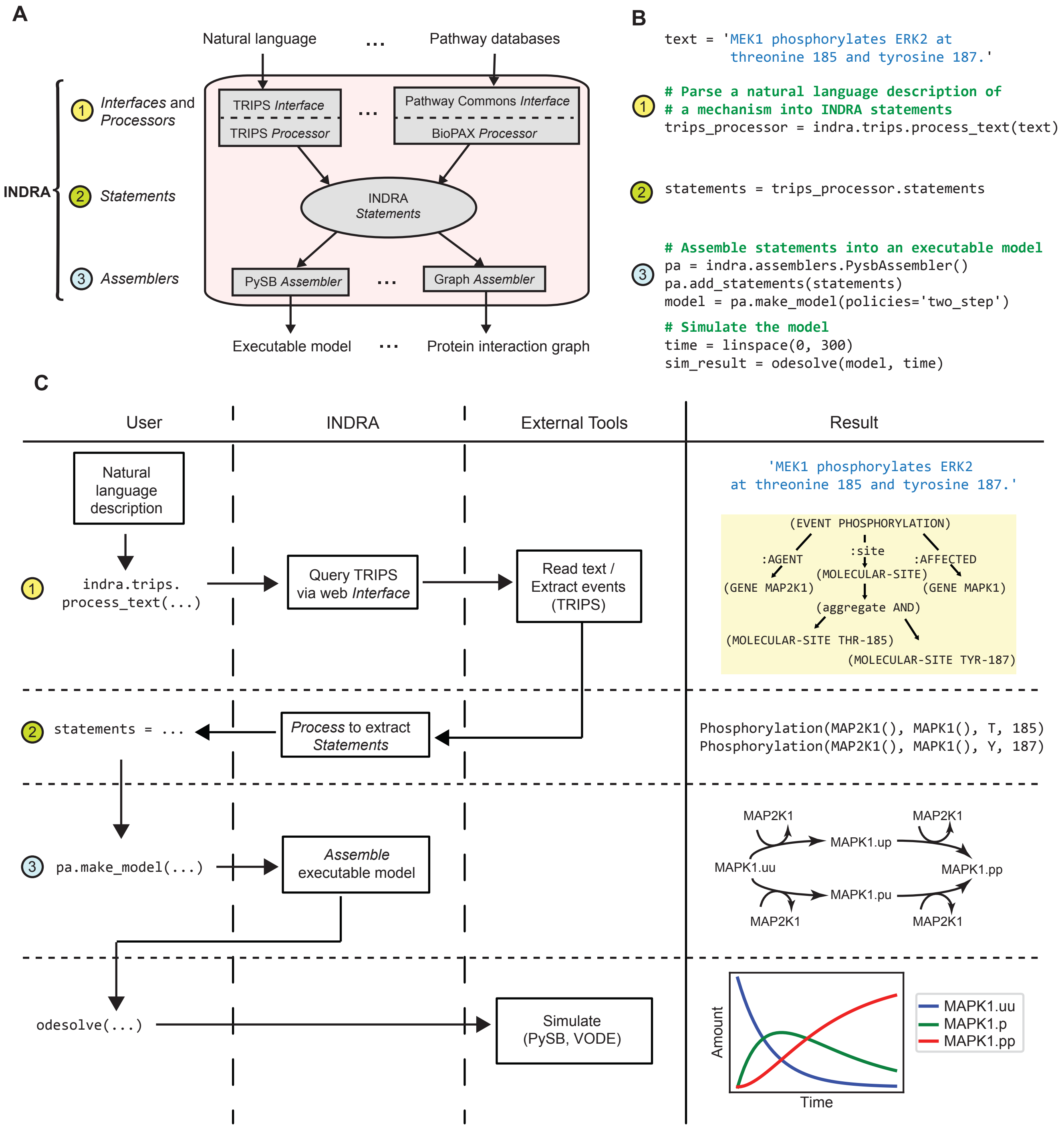
(B) A correction on the pathway map is made by editing the original text. One sentence is removed (red sentence) and is replaced by another one (green sentence) as a basis for the updated assembly whose relevant parts are shown as a graph below. P90RSK is removed as a substrate of mTORC2 and added as a substrate of MAPK1 and MAPK3 (green highlight).

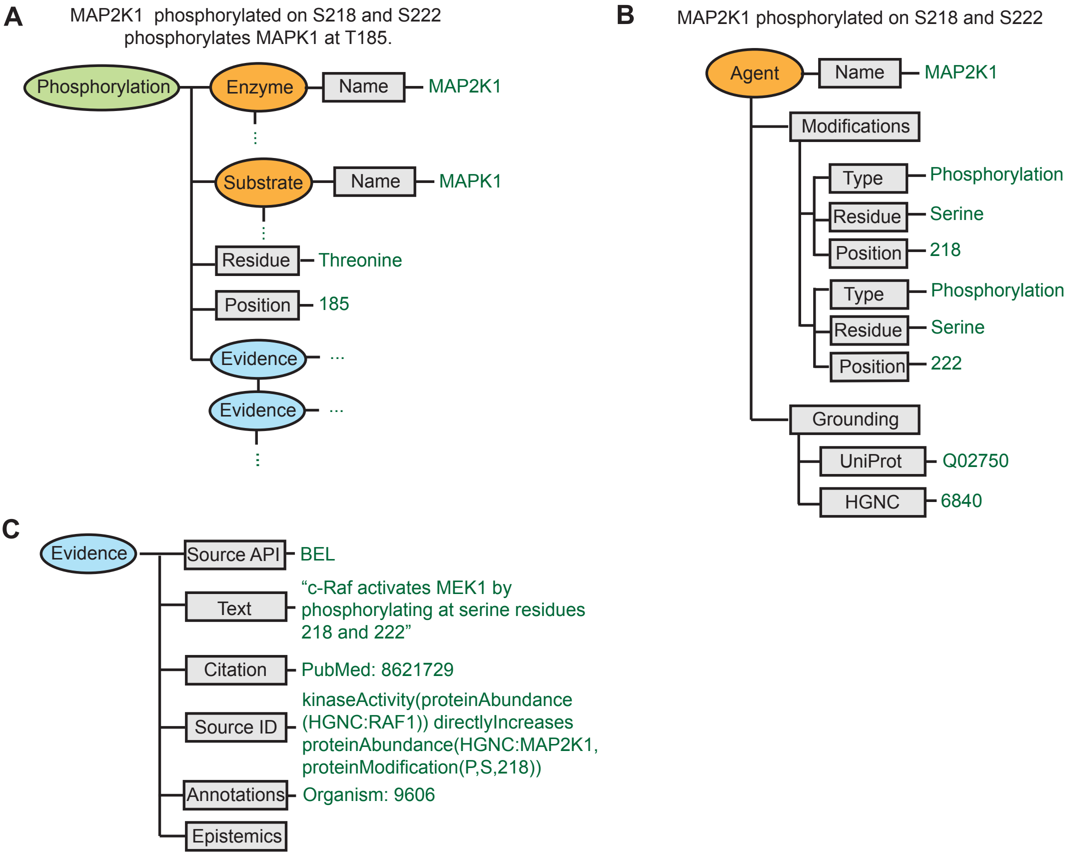
(C) The pathway map is extended with a new branch by adding four additional sentences describing JNK signaling. The newly added pathway (green highlight; gene names appearing as their standard gene symbols, for instance “HPK1” in the original sentences is represented as the node MAP4K1) provides a parallel path from EGFR to the JUN transcription factor, both of which were included in the original model.

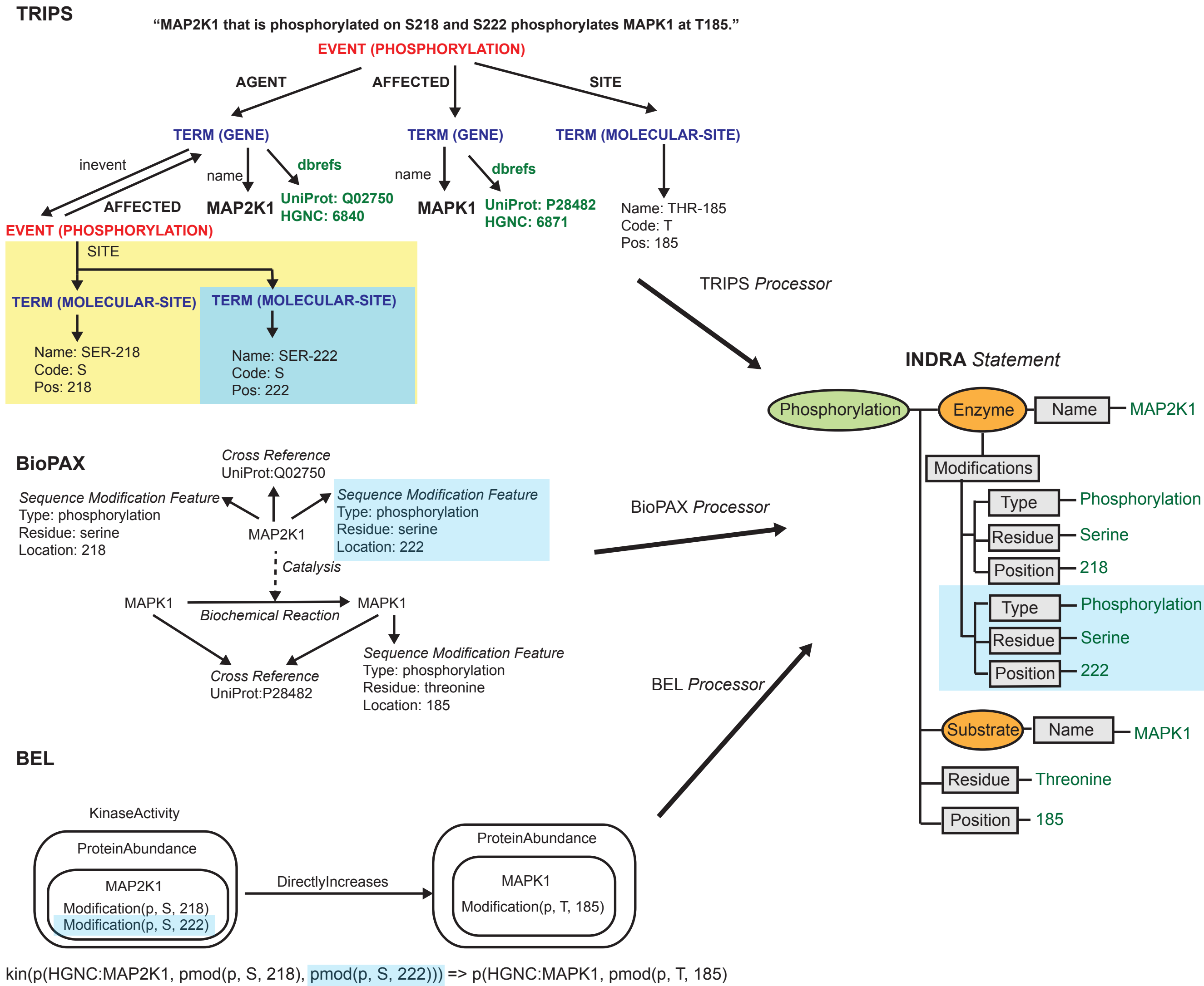
Figure 8. Approaches to building dynamical models of biochemical mechanisms.

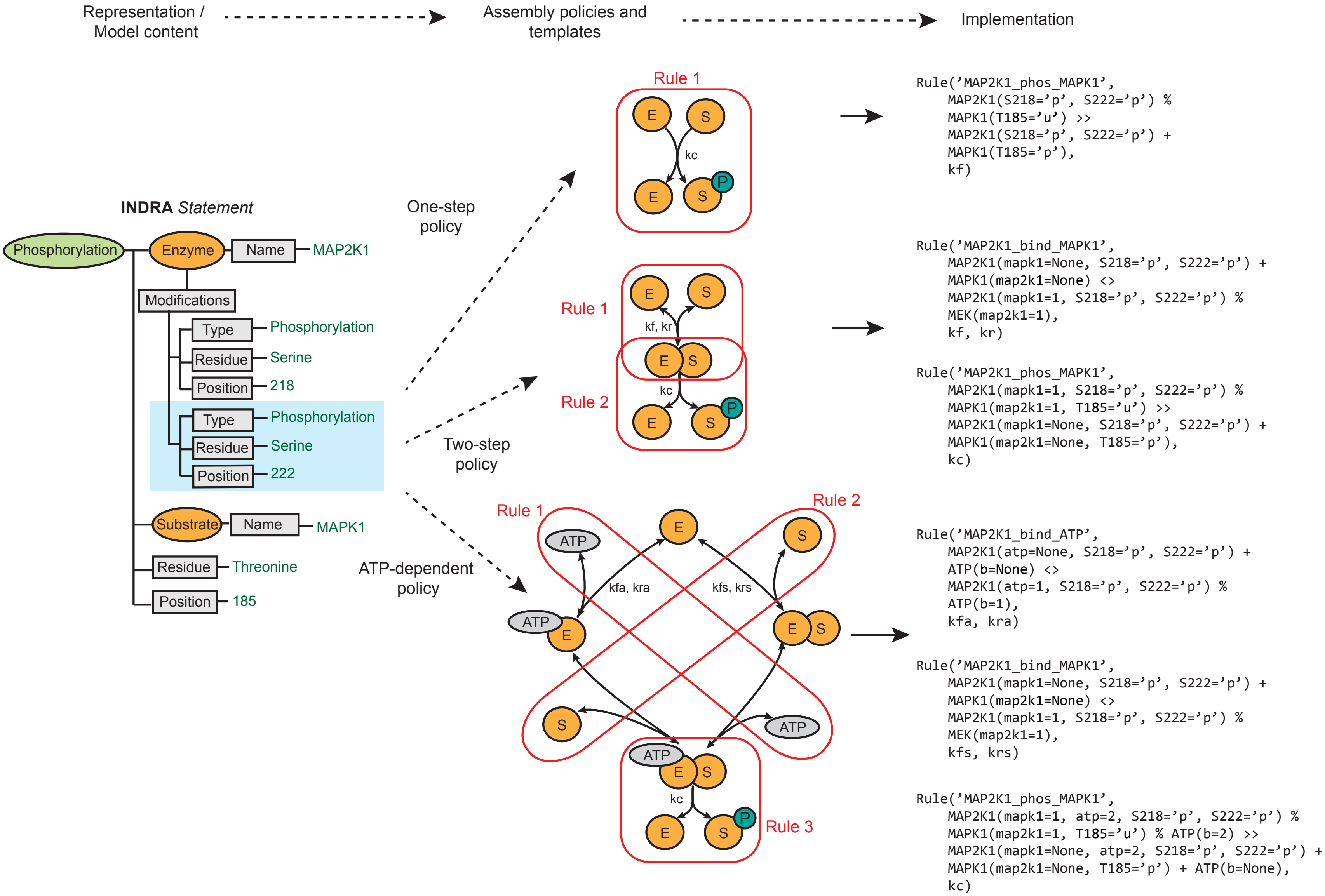
(A) Stages of describing a mechanism from concept to implementation. The mechanism “an enzyme binds a substrate” is shown at different levels of abstraction from mechanistic concept to equation-level implementation. The conceptual description can be expressed in natural language, which can be formalized as an INDRA *Statement* between an enzyme and a substrate *Agent*. The PySB description and a corresponding BioNetGen description (see Box 3) describe a particular implementation of this mechanism in terms of a single rule, which corresponds to a “low-level” instance of three differential equations describing the temporal behavior of the enzyme, substrate and their complex in time.

(B) Comparison of “classical” mathematical modeling (left), programmatic modeling with PySB (center) and modeling with INDRA (right). In each paradigm, red arrows show processes that are done by the user and green arrows show ones that are automatically generated.

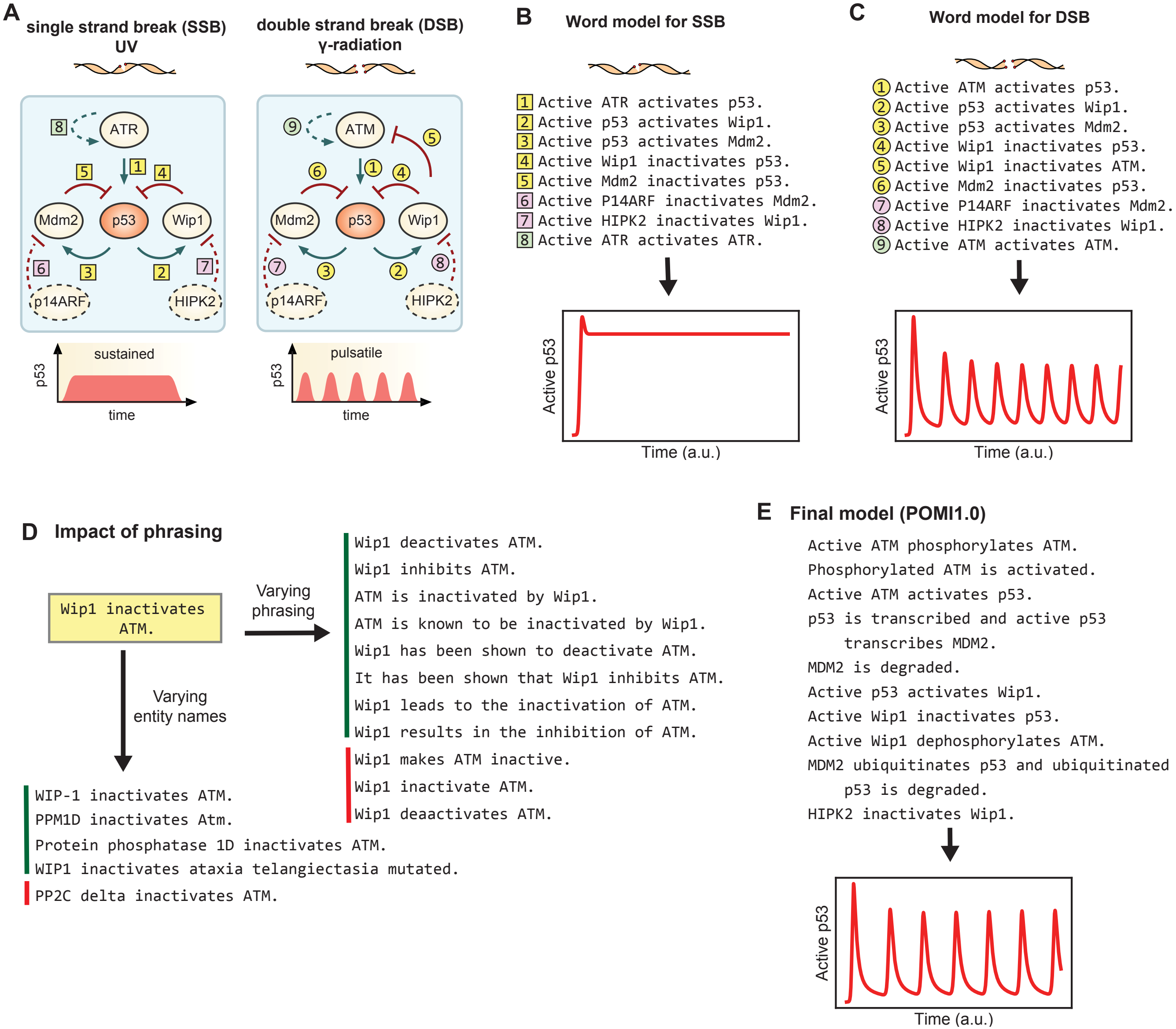

A
Natural language
...
Pathway databases
INDRA
1 Interfaces and Processors
2 Statements
3 Assemblers
TRIPS Interface
TRIPS Processor
Pathway Commons Interface
BioPAX Processor
INDRA Statements
PySB Assembler
Graph Assembler
Executable model
Protein interaction graph
B
text = 'MEK1 phosphorylates ERK2 at threonine 185 and tyrosine 187.'
# Parse a natural language description of
# a mechanism into INDRA statements
trips_processor = indra.trips.process_text(text)
statements = trips_processor.statements
# Assemble statements into an executable model
pa = indra.assemblers.PysbAssembler()
pa.add_statements(statements)
model = pa.make_model(policies='two_step')
# Simulate the model
time = linspace(0, 300)
sim_result = odesolve(model, time)
C
User
INDRA
External Tools
Result
Natural language description
indra.trips.process_text(...)
Query TRIPS via web Interface
Read text / Extract events (TRIPS)
'MEK1 phosphorylates ERK2 at threonine 185 and tyrosine 187.'
(EVENT PHOSPHORYLATION)
:AGENT
:site
:AFFECTED
(MOLECULAR-SITE)
(GENE MAP2K1)
(GENE MAPK1)
(aggregate AND)
(MOLECULAR-SITE THR-185)
(MOLECULAR-SITE TYR-187)
statements = ...
Process to extract Statements
Phosphorylation(MAP2K1(), MAPK1(), T, 185)
Phosphorylation(MAP2K1(), MAPK1(), Y, 187)
pa.make_model(...)
Assemble executable model
MAP2K1
MAPK1.up
MAPK1.uu
MAPK1.pp
MAPK1.pu
odesolve(...)
Simulate (PySB, VODE)
Amount
Time
MAPK1.uu
MAPK1.p
MAPK1.pp


A
MAP2K1 phosphorylated on S218 and S222 phosphorylates MAPK1 at T185.
Phosphorylation
Enzyme
Name
MAP2K1
Substrate
Name
MAPK1
Residue
Threonine
Position
185
Evidence
...
Evidence
...
B
MAP2K1 phosphorylated on S218 and S222
Agent
Name
MAP2K1
Modifications
Type
Phosphorylation
Residue
Serine
Position
218
Type
Phosphorylation
Residue
Serine
Position
222
Grounding
UniProt
Q02750
HGNC
6840
C
Evidence
Source API
BEL
Text
"c-Raf activates MEK1 by phosphorylating at serine residues 218 and 222"
Citation
PubMed: 8621729
Source ID
kinaseActivity(proteinAbundance (HGNC:RAF1)) directlyIncreases proteinAbundance(HGNC:MAP2K1, proteinModification(P,S,218))
Annotations
Organism: 9606
Epistemics

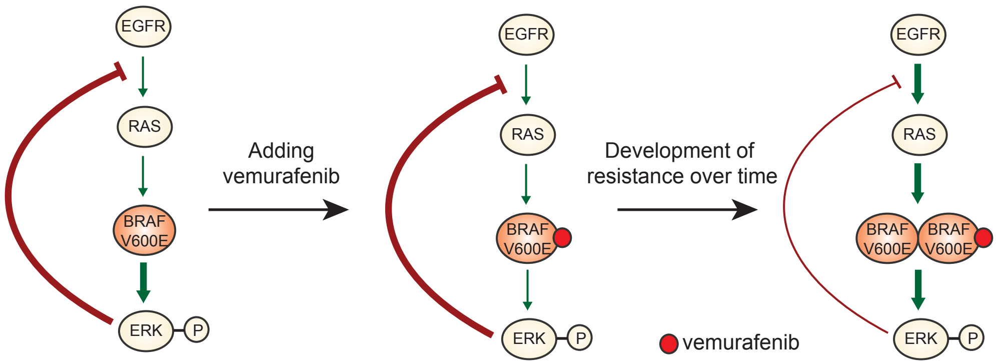

TRIPS
"MAP2K1 that is phosphorylated on S218 and S222 phosphorylates MAPK1 at T185."
EVENT (PHOSPHORYLATION)
AGENT
AFFECTED
SITE
TERM (GENE)
TERM (GENE)
TERM (MOLECULAR-SITE)
inevent
name
dbrefs
name
dbrefs
AFFECTED
MAP2K1
UniProt: Q02750
HGNC: 6840
MAPK1
UniProt: P28482
HGNC: 6871
Name: THR-185
Code: T
Pos: 185
EVENT (PHOSPHORYLATION)
SITE
TERM (MOLECULAR-SITE)
TERM (MOLECULAR-SITE)
Name: SER-218
Code: S
Pos: 218
Name: SER-222
Code: S
Pos: 222
TRIPS Processor
INDRA Statement
Phosphorylation
Enzyme
Name
MAP2K1
Modifications
Type
Phosphorylation
Residue
Serine
Position
218
Type
Phosphorylation
Residue
Serine
Position
222
Substrate
Name
MAPK1
Residue
Threonine
Position
185
BioPAX
Cross Reference
UniProt:Q02750
Sequence Modification Feature
Type: phosphorylation
Residue: serine
Location: 218
MAP2K1
Sequence Modification Feature
Type: phosphorylation
Residue: serine
Location: 222
Catalysis
MAPK1
Biochemical Reaction
MAPK1
Cross Reference
UniProt:P28482
Sequence Modification Feature
Type: phosphorylation
Residue: threonine
Location: 185
BioPAX Processor
BEL Processor
BEL
KinaseActivity
ProteinAbundance
MAP2K1
Modification(p, S, 218)
Modification(p, S, 222)
DirectlyIncreases
ProteinAbundance
MAPK1
Modification(p, T, 185)
kin(p(HGNC:MAP2K1, pmod(p, S, 218), pmod(p, S, 222))) => p(HGNC:MAPK1, pmod(p, T, 185)



Representation / Model content
Assembly policies and templates
Implementation
INDRA Statement
Phosphorylation
Enzyme
Name
MAP2K1
Modifications
Type
Phosphorylation
Residue
Serine
Position
218
Type
Phosphorylation
Residue
Serine
Position
222
Substrate
Name
MAPK1
Residue
Threonine
Position
185
One-step policy
Two-step policy
ATP-dependent policy
Rule 1
E
S
kc
E
S
P
Rule 1
kf, kr
Rule 2
kc
Rule 1
Rule 2
Rule 3
ATP
kfa, kra
kfs, krs
Rule('MAP2K1_phos_MAPK1',
MAP2K1(S218='p', S222='p') %
MAPK1(T185='u') >>
MAP2K1(S218='p', S222='p') +
MAPK1(T185='p'),
kf)
Rule('MAP2K1_bind_MAPK1',
MAP2K1(mapk1=None, S218='p', S222='p') +
MAPK1(map2k1=None) <>
MAP2K1(mapk1=1, S218='p', S222='p') %
MEK(map2k1=1),
kf, kr)
Rule('MAP2K1_phos_MAPK1',
MAP2K1(mapk1=1, S218='p', S222='p') %
MAPK1(map2k1=1, T185='u') >>
MAP2K1(mapk1=None, S218='p', S222='p') +
MAPK1(map2k1=None, T185='p'),
kc)
Rule('MAP2K1_bind_ATP',
MAP2K1(atp=None, S218='p', S222='p') +
ATP(b=None) <>
MAP2K1(atp=1, S218='p', S222='p') %
ATP(b=1),
kfa, kra)
Rule('MAP2K1_bind_MAPK1',
MAP2K1(mapk1=None, S218='p', S222='p') +
MAPK1(map2k1=None) <>
MAP2K1(mapk1=1, S218='p', S222='p') %
MEK(map2k1=1),
kfs, krs)
Rule('MAP2K1_phos_MAPK1',
MAP2K1(mapk1=1, atp=2, S218='p', S222='p') %
MAPK1(map2k1=1, T185='u') % ATP(b=2) >>
MAP2K1(mapk1=None, atp=2, S218='p', S222='p') +
MAPK1(map2k1=None, T185='p') + ATP(b=None),
kc)


A
single strand break (SSB)
UV
double strand break (DSB)
γ-radiation
ATR
Mdm2
p53
Wip1
p14ARF
HIPK2
ATM
sustained
pulsatile
p53
time
B
Word model for SSB
1 Active ATR activates p53.
2 Active p53 activates Wip1.
3 Active p53 activates Mdm2.
4 Active Wip1 inactivates p53.
5 Active Mdm2 inactivates p53.
6 Active P14ARF inactivates Mdm2.
7 Active HIPK2 inactivates Wip1.
8 Active ATR activates ATR.
Active p53
Time (a.u.)
C
Word model for DSB
1 Active ATM activates p53.
2 Active p53 activates Wip1.
3 Active p53 activates Mdm2.
4 Active Wip1 inactivates p53.
5 Active Wip1 inactivates ATM.
6 Active Mdm2 inactivates p53.
7 Active P14ARF inactivates Mdm2.
8 Active HIPK2 inactivates Wip1.
9 Active ATM activates ATM.
Active p53
Time (a.u.)
D
Impact of phrasing
Wip1 inactivates ATM.
Varying phrasing
Varying entity names
Wip1 deactivates ATM.
Wip1 inhibits ATM.
ATM is inactivated by Wip1.
ATM is known to be inactivated by Wip1.
Wip1 has been shown to deactivate ATM.
It has been shown that Wip1 inhibits ATM.
Wip1 leads to the inactivation of ATM.
Wip1 results in the inhibition of ATM.
Wip1 makes ATM inactive.
Wip1 inactivate ATM.
Wip1 deaactivates ATM.
WIP-1 inactivates ATM.
PPM1D inactivates Atm.
Protein phosphatase 1D inactivates ATM.
WIP1 inactivates ataxia telangiectasia mutated.
PP2C delta inactivates ATM.
E
Final model (POMI1.0)
Active ATM phosphorylates ATM.
Phosphorylated ATM is activated.
Active ATM activates p53.
p53 is transcribed and active p53 transcribes MDM2.
MDM2 is degraded.
Active p53 activates Wip1.
Active Wip1 inactivates p53.
Active Wip1 dephosphorylates ATM.
MDM2 ubiquitinates p53 and ubiquitinated p53 is degraded.
HIPK2 inactivates Wip1.
Active p53
Time (a.u.)

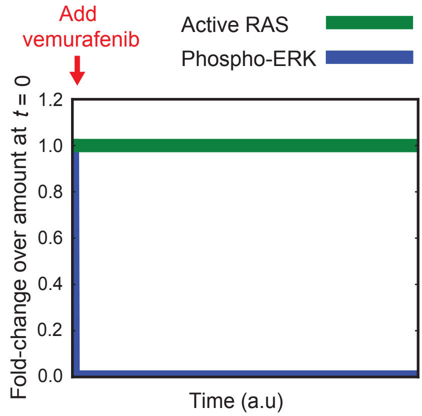
A



B MEMI 1.0

1. The growth factor ligand EGF binds EGFR.
2. The EGFR-EGF complex binds another EGFR-EGF complex.
3. The EGFR-EGFR complex binds GRB2.
4. EGFR-bound GRB2 binds SOS.
5. GRB2-bound SOS binds RAS that is not bound to BRAF V600E.
6. SOS-bound RAS binds GTP.
7. GTP-bound RAS that is not bound to SOS binds BRAF V600E.
8. Vemurafenib binds BRAF V600E.
9. BRAF V600E that is not bound to vemurafenib phosphorylates MEK.
10. PP2A-alpha dephosphorylates MEK that is not bound to ERK.
11. Phosphorylated MEK is activated.
12. Active MEK that is not bound to PP2A-alpha phosphorylates ERK.
13. Phosphorylated ERK is activated.
14. DUSP6 dephosphorylates ERK.

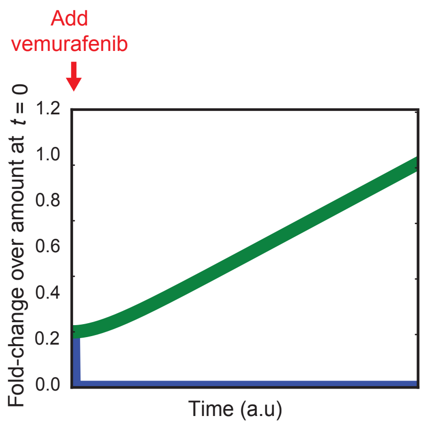
28 rules
41 parameters
99 ODEs



C MEMI 1.1 = MEMI 1.0 + ERK-SOS feedback

4. EGFR-bound GRB2 binds SOS that is not phosphorylated on a serine.
5. GRB2-bound SOS that is not phosphorylated on serine binds RAS that is not bound to BRAF V600E.
14. DUSP6 dephosphorylates ERK that is not bound to SOS.
15. Active ERK that is not bound to DUSP6 phosphorylates on serine a SOS that is not bound to RAS.
16. A phosphatase dephosphorylates SOS on serine.

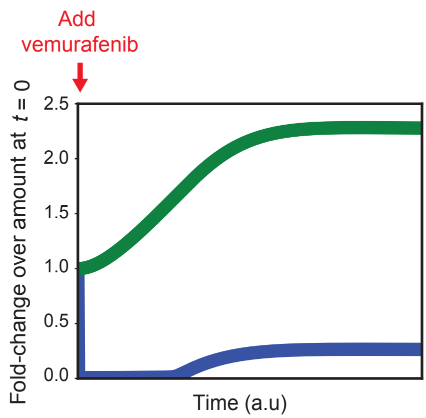
34 rules
50 parameters
275 ODEs

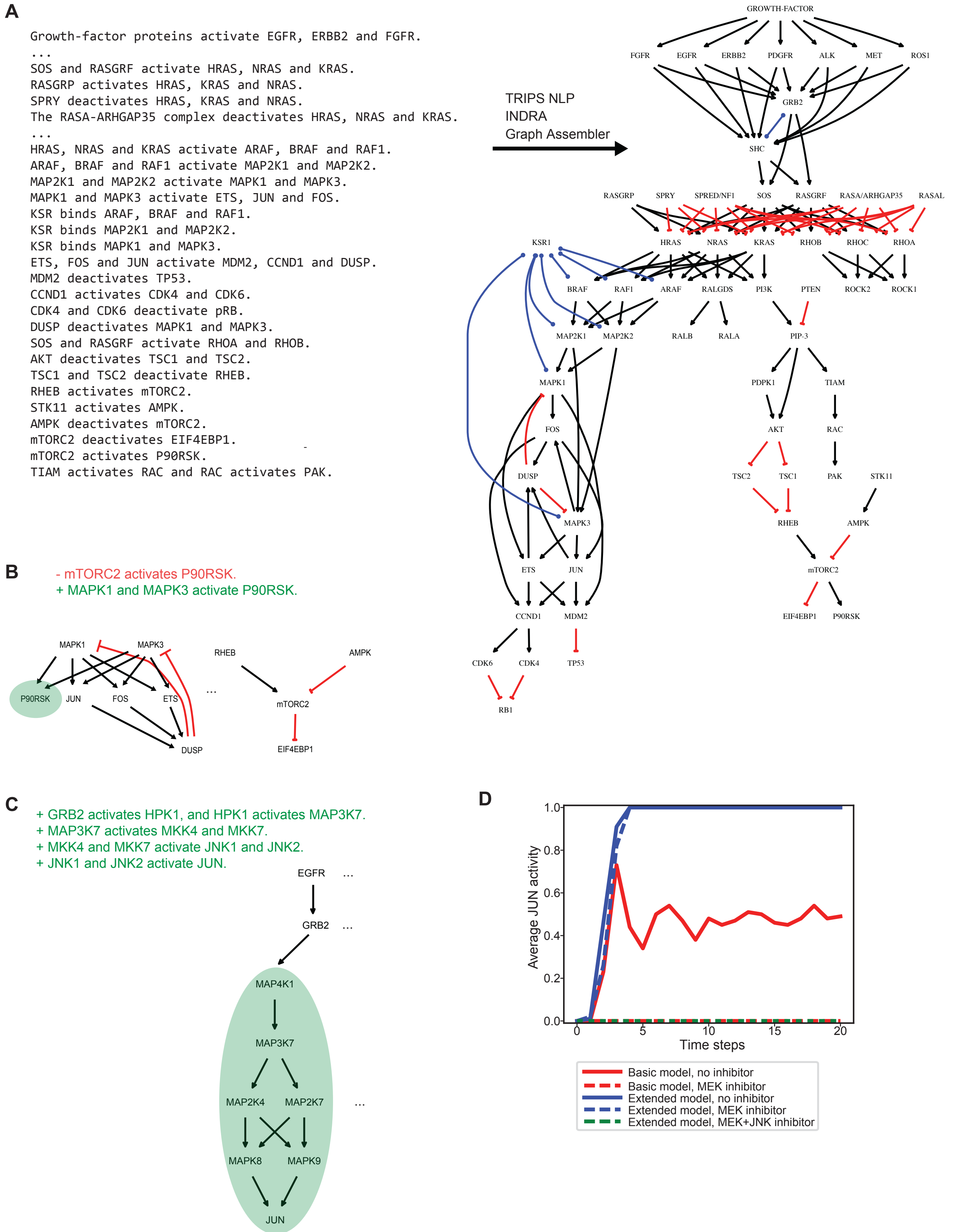


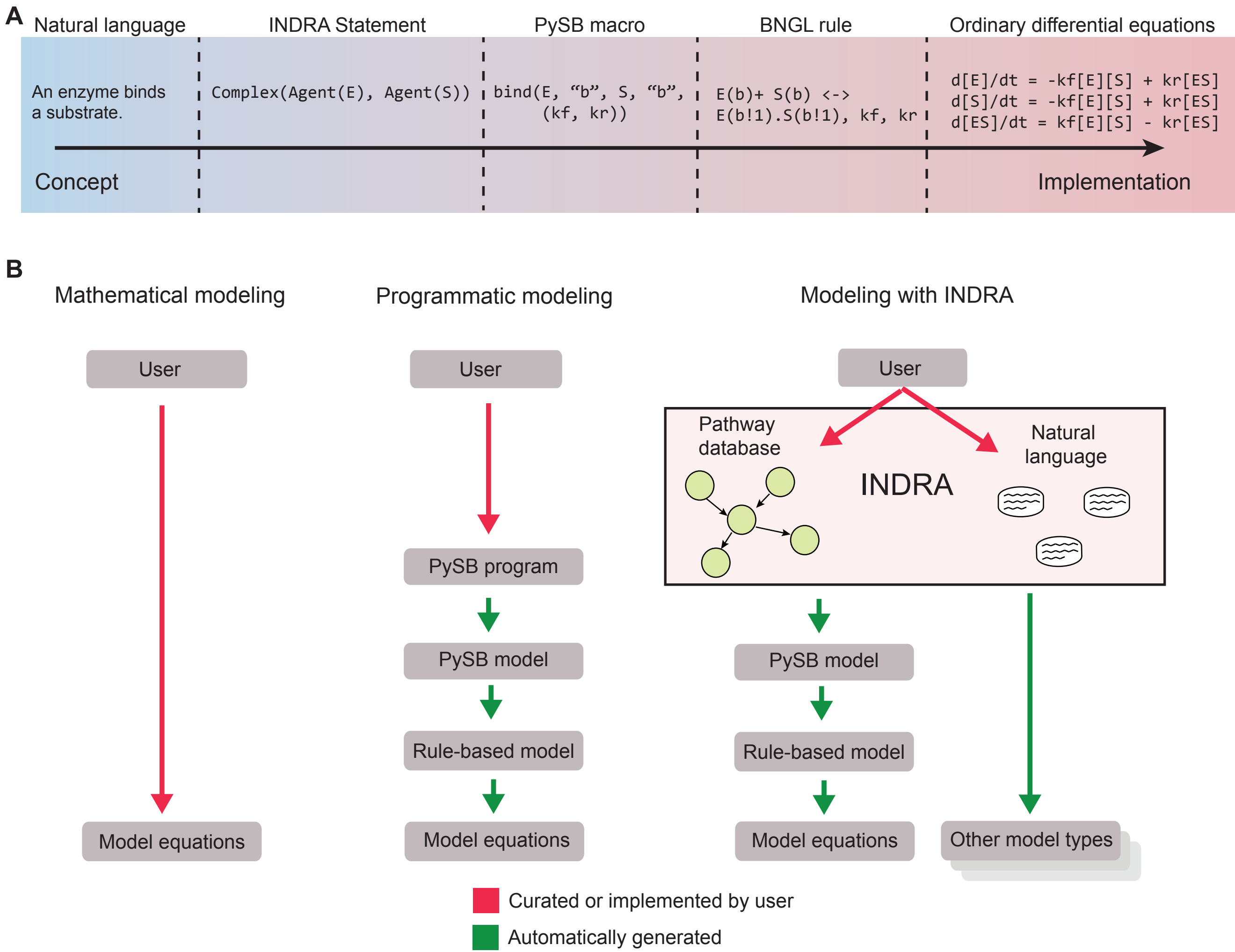
D MEMI 1.2 = MEMI 1.1 + BRAF dimerization

17. RAS-bound BRAF V600E binds RAS-bound BRAF V600E.
8A. Vemurafenib binds BRAF V600E that is not bound to BRAF V600E.
8B. Vemurafenib binds BRAF V600E that is bound to BRAF V600E.

37 rules
54 parameters
353 ODEs




A
Growth-factor proteins activate EGFR, ERBB2 and FGFR.
...
SOS and RASGRF activate HRAS, NRAS and KRAS.
RASGRP activates HRAS, KRAS and NRAS.
SPRY deactivates HRAS, KRAS and NRAS.
The RASA-ARHGAP35 complex deactivates HRAS, NRAS and KRAS.
...
HRAS, NRAS and KRAS activate ARAF, BRAF and RAF1.
ARAF, BRAF and RAF1 activate MAP2K1 and MAP2K2.
MAP2K1 and MAP2K2 activate MAPK1 and MAPK3.
MAPK1 and MAPK3 activate ETS, JUN and FOS.
KSR binds ARAF, BRAF and RAF1.
KSR binds MAP2K1 and MAP2K2.
KSR binds MAPK1 and MAPK3.
ETS, FOS and JUN activate MDM2, CCND1 and DUSP.
MDM2 deactivates TP53.
CCND1 activates CDK4 and CDK6.
CDK4 and CDK6 deactivate pRB.
DUSP deactivates MAPK1 and MAPK3.
SOS and RASGRF activate RHOA and RHOB.
AKT deactivates TSC1 and TSC2.
TSC1 and TSC2 deactivate RHEB.
RHEB activates mTORC2.
STK11 activates AMPK.
AMPK deactivates mTORC2.
mTORC2 deactivates EIF4EBP1.
mTORC2 activates P90RSK.
TIAM activates RAC and RAC activates PAK.
TRIPS NLP
INDRA
Graph Assembler
GROWTH-FACTOR
FGFR EGFR ERBB2 PDGFR ALK MET ROS1
GRB2
SHC
RASGRP SPRY SPRED/NF1 SOS RASGRF RASA/ARHGAP35 RASAL
KSR1 HRAS NRAS KRAS RHOB RHOC RHOA
BRAF RAF1 ARAF RALGDS PI3K PTEN ROCK2 ROCK1
MAP2K1 MAP2K2 RALB RALA PIP-3
MAPK1 PDPK1 TIAM
FOS AKT RAC
DUSP TSC2 TSC1 PAK STK11
MAPK3 RHEB AMPK
ETS JUN mTORC2
CCND1 MDM2 EIF4EBP1 P90RSK
CDK6 CDK4 TP53
RB1
B
- mTORC2 activates P90RSK.
+ MAPK1 and MAPK3 activate P90RSK.
MAPK1 MAPK3 RHEB AMPK
P90RSK JUN FOS ETS ... mTORC2
DUSP EIF4EBP1
C
+ GRB2 activates HPK1, and HPK1 activates MAP3K7.
+ MAP3K7 activates MKK4 and MKK7.
+ MKK4 and MKK7 activate JNK1 and JNK2.
+ JNK1 and JNK2 activate JUN.
EGFR ...
GRB2 ...
MAP4K1
MAP3K7
MAP2K4 MAP2K7 ...
MAPK8 MAPK9
JUN
D
Average JUN activity
1.0
0.8
0.6
0.4
0.2
0.0
0 5 10 15 20
Time steps
Basic model, no inhibitor
Basic model, MEK inhibitor
Extended model, no inhibitor
Extended model, MEK inhibitor
Extended model, MEK+JNK inhibitor


A
Natural language
INDRA Statement
PySB macro
BNGL rule
Ordinary differential equations
An enzyme binds a substrate.
Complex(Agent(E), Agent(S))
bind(E, "b", S, "b", (kf, kr))
E(b)+ S(b) <-> E(b!1).S(b!1), kf, kr
d[E]/dt = -kf[E][S] + kr[ES]
d[S]/dt = -kf[E][S] + kr[ES]
d[ES]/dt = kf[E][S] - kr[ES]
Concept
Implementation
B
Mathematical modeling
Programmatic modeling
Modeling with INDRA
User
User
User
Pathway database
INDRA
Natural language
PySB program
PySB model
PySB model
Rule-based model
Rule-based model
Model equations
Model equations
Model equations
Other model types
Curated or implemented by user
Automatically generated