Comprehensive benchmark dataset of real-world Boolean network models

Samuel Pastva[1,2], David Šafránek[2], Nikola Beneš[2], and Luboš Brim[2]

[1]Institute of Science and Technology Austria, Klosterneuburg, 3400, Austria
[2]Faculty of Informatics, Masaryk University, Brno, 60200, Czechia
[*]corresponding author: Samuel Pastva (samuel.pastva@ist.ac.at; xpastva@fi.muni.cz)

ABSTRACT

Recent developments in both computational analysis and data-driven syntesis enable a new era of automated reasoning with logical models (Boolean networks in particular) in systems biology. However, these advancements also motivate an increased focus on quality control and scalability comparisons between tools.
At the moment, to illustrate real-world applicability, authors typically test their approaches on small sets of manually curated models that are inherently limited in scope. This further complicates reuse and comparisons, because benchmark models often contain ad hoc modifications or are outright not available.
In this paper, we describe a new, comprehensive, open source dataset of 210+ Boolean network models compiled from available databases and a literature survey. The models are available in a wide range of formats. Furthermore, the dataset is accompanied by a validation pipeline that ensures the integrity and logical consistency of each model. Using this pipeline, we identified and repaired many potential problems in a number of widely used models.

Background & Summary

Logical models in general, and Boolean networks in particular, represent one of the fundament modelling frameworks employed by systems biology[1]. Historically, much of the Boolean network design and analysis has been performed either manually, or through computer assisted approaches requiring extensive involvement of domain experts[2–4]. However, recent developments in topics like attractor[5–7] and trap space[8] detection, bifurcation analysis[9], and synthesis from data and prior knowledge[10,11] enable comprehensive automated analysis of large, possibly partially unknown networks comprising hundreds of components.

This proliferation of Boolean network tools and methods motivates the need for extensive and reproducible validation using realistic real-world models. Typically, this role is served by ad hoc subsets of models available in popular systems biology databases such as CellCollective[12], Biomodels[13] or GINsim[14]. However, this approach suffers from many obvious downsides.

While these databases have non-trivial overlap, they are mostly focused on specific human-curated models. As such, they are inherently limited in scope to the models recognised by the database curators. Furthermore, models available from multiple sources often contain subtle but critical modifications in the network structure and initial conditions. Finally, these sources often lack automated validation of the model integrity and logical consistency, meaning errors can propagate to models derived in the future, or go unnoticed when performing analysis.

To address these issues, we present the *Biodivine Boolean Models* (BBM) dataset. BBM covers 210+ models from an extensive set of sources, including the aforementioned databases and independently published models (see Figure 1). For each model, BBM tracks the primary source of the model (i.e. its publication and authors), any secondary sources (e.g. databases) and the source for the actual model files (e.g. git repositories, supplementary materials, database entries, etc.).

Aside from model aggregation, BBM provides several benefits to tool authors: First, models tracked in BBM are available in several widely recognised formats (`bnet`[15], `aeon`[16], and `sbml`[17]) and can be downloaded in bulk. Second, BBM eliminates certain ambiguities in model representation, such as the treatment of network input nodes. Third, BBM performs extensive static validation to ensure logical consistency between variable update rules and the associated regulatory network of each model. This process identifies and repairs common problems in logical models such as mismatch in monotonicity of a regulation and an update function, or the presence of redundant model components. Finally, BBM provides an automated reproducible workflow to generate custom variants of the dataset based on the user provided criteria.

Overall, we believe BBM significantly simplifies the process of validating and benchmarking Boolean network tools as it provides a large, clearly defined and validated set of models. To the best of our knowledge, BBM is also the largest publicly available set of real-world (i.e. not randomly generated) Boolean network models.

Methods

In this section, we first give a quick overview of the relevant elements of Boolean network model. We then proceed to list how the actual BBM dataset is created, curated, validated and distributed.

Boolean networks

First, let us give a brief formal introduction into the topic of Boolean network modelling. By $\mathbb{B}$, we denote the set $\{0,1\}$. We then say that a *Boolean network* $\mathscr{B}$ over n variables is an indexed collection $\mathscr{B} = \{F_1, \dots, F_n\}$, such that each F_i is a Boolean *update function* $F_i : \mathbb{B}^n \to \mathbb{B}$ (the set of Boolean vectors $\mathbb{B}^n$ is called the *state space* of $\mathscr{B}$). Naturally, within a real-world model, variables are also augmented with human-readable names and other biologically relevant metadata, while functions are described using Boolean expressions or function tables.

Signed Influence graph Individual update functions typically depend only on a small subset of network variables, not the whole network state. This fact is communicated using the *signed influence graph* (sometimes called the *regulatory network*) associated with the Boolean network $\mathscr{B}$. This is a labelled directed graph $\mathscr{I} = (V, E, l)$ such that its vertices are the variables of $\mathscr{B}$, $V = \{1, \dots, n\}$, edges $E \subseteq V \times V$ represent dependencies between variables (we say that x *regulates* y when $(x,y) \in E$) and $l : E \to \{\oplus, \ominus, ?\}$ labels each edge with an optional *monotonicity* (positive, negative, or unknown). The influence graph is typically provided directly by model authors.

Properties of Boolean functions Intuitively, the influence graph $\mathscr{I}$ imposes additional requirements on the structure of the network's update functions F_i. Formally, we can understand these requirements as *essentiality* and *monotonicity* of individual function inputs. First, given a state $x \in \mathbb{B}^n$, we write $x[i \leftarrow b]$ to denote a copy of x with the value of the i-th variable substituted for $b \in \mathbb{B}$. Then, given a function $F : \mathbb{B}^n \to \mathbb{B}$, we say that the i-th variable is an *essential* input of F if the output of F depends on this variable, formally $\exists x \in \mathbb{B}^n.\ F(x[i \leftarrow 0]) \neq F(x[i \leftarrow 1])$. We also say that F is *positively monotonic* in its i-th input when increasing this input cannot decrease the output of F, formally $\forall x \in \mathbb{B}.\ F(x[i \leftarrow 0]) \leq F(x[i \leftarrow 1])$. Symmetrically, we say that the input is *negatively monotonic* when $\forall x \in \mathbb{B}.\ F(x[i \leftarrow 0]) \geq F(x[i \leftarrow 1])$.

In practice, these essentiality and monotonicity requirements map to the biological assumptions about the modelled system. For example, if a transcription factor is observed to activate the expression of a particular gene in a wet lab experiment, we expect it to act as an essential and positively monotonic input in the corresponding update function of our in silico model. Overall, we expect that the essential inputs of each update function are exactly the regulators declared within the influence graph and that the monotonicity of these inputs matches the associated edge labels l.

Model Acquisition

Next, we describe process of compiling the BBM dataset. Due to the high variability of model sources, the process of incorporating a new model is largely manual (aside from validation and consistency checking covered in the next section).

Eligible models To cover the widest possible assortment of realistic models, we currently allow submission of any logical model that can be reliably retrieved from a known database, repository, or an associated publication. The only requirement is that the model must be based on a real biological system. That is, we do not accept randomly generated or otherwise synthetic models. However, we do not require any specific level of curation or validation of any biological claims. In particular, the model can be hand made, automatically synthesised, or anything in between. Similarly, we accept many different model formats and we are willing to assist with translation if a machine-friendly model representation is not available.

Multi-valued networks The dataset also incorporates multi-valued networks. That is, logical models admitting multi-valued or bounded integer variables (as opposed to just Booleans). To maximise compatibility with the available tools, every such model is instead provided in a Booleanised representation. That is, the model is transformed into a plain Boolean network such that each multi-valued component with domain $\{0, \dots, k\}$ is expanded into k Boolean components using the well-known Van Ham encoding[18]. This process is facilitated by the tool `biolqm`[19]. Each Booleanised multi-valued model is distinguished using a `multi-valued` keyword included in model's metadata. Whenever possible, we also incorporate the `sbml`[17] representation of the original (i.e. non-Booleanised) multi-valued model. However, such original files currently cannot be processed by the automated validation pipeline.

Sources The dataset incorporates 219 models from a wide range of publications. Notable sources include GINsim[14] (83 models), CellCollective[12] (78 models), Biomodels[13] (18 models) and the COVID-19 Disease Map[20] (22 models) databases. If a model appears in multiple databases, the dataset lists such model only once with multiple data source entries, even if there are minor differences between the model files in each database. The contribution of each database to the overall dataset is visualised in Figure 1. Furthermore, Online-only Table 1 gives the full list of models including references to their original publications and Online-only Table 2 maps relevant database entries to the corresponding BBM models. The remaining models are sourced through a comprehensive literature search for Boolean network related keywords across PubMed and Google Scholar.

Normalisation, Validation and Repair

Within BBM, each model is initially assigned a unique integer identifier (ID) and a human-readable name. Additionally, an *input* model file is required, using one of the eligible formats: `bnet`[15], `aeon`[16], and `sbml`[17]. In some cases, such representation is not available. For example, some models are described directly within the text of a supplementary material of a publication. For such models, we perform a manual conversion into one of the supported formats, but we also maintain the original model representation for future reference.

Our automated pipeline then translates the input model file into all three formats and performs several normalisation steps to maximise compatibility with other tools. First, we ensure that all variable names are unique and do not contain any special characters (in particular for `sbml`, model variables are further distinguished by separate IDs, hence a name can be an arbitrary string). Then, several normalisation and validation steps are performed to ensure the model contains no additional issues.

Input node normalisation A variable of a Boolean network is typically called an *input* if it has no incoming regulations (no incoming edges within the influence graph). Consequently, update function of such variable can be only *true* or *false*, since it cannot have any essential inputs. The particular choice of update functions then constitutes the *initial conditions* of the model. However, some authors also consider inputs to be constant but unknown, meaning their update function is the identity function. In such case, the variable has a positive self-regulation, but this is sometimes omitted if clear from context. The initial conditions of the model (if any) then need to be provided separately. Finally, some tools also allow to omit the update function of input nodes entirely, giving yet another possible representation for constant but unknown inputs.

The distinction between these representations is subtle but crucial in many instances: For example, when considering attractor detection, a constant *true* or *false* restricts the search to the specified value. Meanwhile, if the input is constant but unknown, the result covers attractors for every possible combination of input values. The number of attractors thus increases exponentially with the number of inputs in such a scenario.

In our pipeline, we detect the presence of network input nodes and normalise them during export. By default, models are distributed without the update functions on all recognised input nodes. However, using the custom dataset export procedure (discussed in the later *Data Distribution* section), users can chose what representation of inputs best suits their needs (i.e. a constant value, identity update function, or no update function at all).

Regulation monotonicity and essentiality Another step in our pipeline is responsible for validating the declared influence graph of the network against the actual update functions employed by the model. We do this using the symbolic BDD framework within the tool `aeon.py`[21]. This procedure allows us to formally verify the update function properties as described in the *Boolean networks* section. If an inconsistency is found, we manually adjust the influence graph such that the problem is removed, but we note this change in the metadata of the model. As such, the actual Boolean dynamics of the model are unchanged, but the error is recorded for future reference. The following errors can be detected and repaired using this procedure:

1. Input j is essential in F_i, but there is no edge $(j,i) \in E$ in the influence graph; repaired by adding this regulation edge.

2. Regulation $(j,i) \in E$ appears in the influence graph, but j is not an essential input of F_j; repaired either by removing the regulation edge, or by annotating the regulation as *non-essential* (only some output formats support such annotation).

3. Regulation $(j,i) \in E$ has a known monotonicity $l(j,i) \in \{\oplus, \ominus\}$, but the input j of F_i does not exhibit this monotonicity; repaired by setting the monotonicity label to ? (unknown).

Note that in some instances, these issues may be intentional. For example, if an error of type 2 is detected such that the update function expression does not contain the incriminated variable at all, this could be a mechanism by which the authors specify that the relationship between the two variables is notable, but unused in this particular system. In such case, the *non-essential* annotation is the correct way to resolve this issue.

However, for the vast majority of errors we observed, this is not the case: The type 2 and 3 errors arose primarily from update functions that syntactically contain the problematic variable, but due to the structure of the function, either the variable does not contribute towards the output of the function at all, or the monotonicity of this contribution does not match the monotonicity declared by the authors.

Unused components Finally, we observe that many models admit variables that do not interact with the rest of the model in any meaningful way, or that such variables appear once all input essentiality errors are corrected. In general, we thus validate that the influence graph of the network in question is always weakly connected. That is, between any two variables, there must be some form of a direct or transitive relationship. If such disconnected components are found, we only retain the largest weakly connected component.

Dataset Distribution

The dataset is managed through a versioned git repository that is continuously updated with new models. However, we regularly publish official *editions* of the dataset which freeze the set of models at a particular time-point and can be referenced separately.

Model metadata As already discussed, each model within a particular BBM edition is available in three formats, `bnet`[15], `aeon`[16], and `sbml`[17], and with erased update functions on input variables. Furthermore, each model is accompanied by a collection of metadata which is provided both as a machine-friendly `json` file and a human-readable `markdown` document. As part of this metadata, we track the publication where the model first appeared (as a BibTeX reference), the original source of the model file, description of any modifications made to the model during validation, as well as a set of relevant keywords.

Using these keywords, we track the databases in which the model appears (e.g. `cell-collective`, `ginsim`, etc.) or methods used to construct the model (e.g. `casq`[22]). We also use the keyword `repaired` to label models where some logical inconsistencies were found during validation (the specific nature of the repair performed is detailed as a separate note), and keyword `multi-valued` to indicate that the model is a Booleanised version of a multi-valued network.

Custom datasets We recognise that our chosen model representation may not be suitable for all benchmarking and validation tasks. As such, we also include an interactive workflow which allows anyone to export a custom edition based on the BBM dataset. Specifically, this workflow allows the user to:

- chose between the three model formats;
- chose the desired representation of input variables (constant *true*, constant *false*, identity function, erased function);
- filter the results using a custom Python expression evaluated for each model and its metadata.

Using this filter functionality, users can pick models based on structural properties (presence of a specific variable or regulation, the number of variables, connectivity, etc.), but also based on model metadata (model author, keywords, etc.). To ensure reproducibility, each such custom dataset edition incorporates an additional metadata file with all parameters of the workflow, meaning the custom edition can be fully recreated based on this metadata file.

Data Records

There are currently two editions of the BBM dataset, both available as Zenodo archives: edition 2021 TODO and edition 2022 TODO. Any future updates to the dataset will be announced using the git repository at https://github.com/sybila/biodivine-boolean-models. Both editions contain the raw input model files, the generated output models and their metadata, as well as any programs necessary to validate the integrity of the models and to generate custom model editions. The models available in the latest edition (2022) are summarised in the Online-only Table 1.

Technical Validation

Naturally, to ensure data integrity, we validate the syntax of each model file using the tool `aeon.py`[21] (some of the input model files even contain minor syntactic errors that had to be corrected). Furthermore, we already described the individual validation steps performed within our pipeline as part of the *Methods* section. This validation pipeline is executed for each commit of the aforementioned git repository, meaning no errors can be introduced into the dataset.

However, we would like to emphasise the importance of this validation step by analysing the number and types of errors uncovered using this validation pipeline.

Add figure and a paragraph which breaks down the types of errors encountered, etc.

Usage Notes

The model files distributed within BBM are compatible with many common tools for Boolean network analysis. In particular, this includes (but is not limited to) the tools available as part of the CoLoMoTo Jupyter notebook environment.[23] Specifically, one can use the following tools (in alphabetical order) to further analyse the properties of the published models:

- `aeon`[16] and `aeon.py`[21] perform symbolic analysis of asynchronous network dynamics (attractors, fixed-points, reachability) and bifurcation analysis with respect to network inputs;
- `biolqm`[19] and `ginsim`[14] facilitate analysis of multi-valued models and other miscellaneous methods (translation between additional model formats, simulation, perturbation analysis, etc.);

- `cabean`[24] facilitates target control and reprogramming of network behaviour;
- `maboss`[25] supports analysis of BNs under *stochastic* update semantics;
- `mpbn`[26] computes the attractors and reachability properties in the *most-permissive* BN semantics;
- `pyboolnet`[15] supports generation of random networks and synchronous simulation;
- `pystablemotifs`[27] uses succession diagrams for attractor computation and network reprogramming;
- `trappist`[8] computes the minimal and maximal trap spaces of the network as well as fixed-points.

For further reading, we recommend the repository of the CoLoMoTo notebook environment, which contains usage examples for most of the aforementioned tools distributed as user-friendly Python notebooks.

Code availability

The code, including previous revisions and all relevant model source files, is available as a git repository at the URL https://github.com/sybila/biodivine-boolean-models. Historical revisions of the source code associated with each BBM edition are also part of the aforementioned Zenodo packages. Submissions of new models are encouraged through issues and pull requests of the git repository (see instructions in the `CONTRIBUTING.md` file). The automated validation and post-processing relies on the tool `aeon.py`[21], version `1.1.0`. Furthermore, `biolqm`[19] was manually used for some of the initial model conversions, but is not used by the automated validation and export pipeline.

References

1. Schwab, J. D., Kühlwein, S. D., Ikonomi, N., Kühl, M. & Kestler, H. A. Concepts in boolean network modeling: What do they all mean? *Comput. structural biotechnology journal* **18**, 571–582 (2020).

2. Zhang, R. *et al.* Network model of survival signaling in large granular lymphocyte leukemia. *Proc. Natl. Acad. Sci.* **105**, 16308–16313 (2008).

3. Fauré, A., Naldi, A., Chaouiya, C. & Thieffry, D. Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics* **22**, e124–e131 (2006).

4. Davidich, M. I. & Bornholdt, S. Boolean network model predicts cell cycle sequence of fission yeast. *PloS one* **3**, e1672 (2008).

5. Beneš, N., Brim, L., Pastva, S. & Šafránek, D. Computing bottom SCCs symbolically using transition guided reduction. In *Computer Aided Verification: 33rd International Conference, CAV 2021, Virtual Event, July 20–23, 2021, Proceedings, Part I 33*, 505–528 (Springer, 2021).

6. Van Giang, T. & Hiraishi, K. An improved method for finding attractors of large-scale asynchronous boolean networks. In *2021 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, 1–9 (IEEE, 2021).

7. Rozum, J. C., Gómez Tejeda Zañudo, J., Gan, X., Deritei, D. & Albert, R. Parity and time reversal elucidate both decision-making in empirical models and attractor scaling in critical boolean networks. *Sci. Adv.* **7**, eabf8124 (2021).

8. Trinh, V.-G., Benhamou, B., Hiraishi, K. & Soliman, S. Minimal trap spaces of logical models are maximal siphons of their petri net encoding. In *Computational Methods in Systems Biology: 20th International Conference, CMSB 2022, Bucharest, Romania, September 14–16, 2022, Proceedings*, 158–176 (Springer, 2022).

9. Beneš, N., Brim, L., Pastva, S., Poláček, J. & Šafránek, D. Formal analysis of qualitative long-term behaviour in parametrised boolean networks. In *Formal Methods and Software Engineering: 21st International Conference on Formal Engineering Methods, ICFEM 2019, Shenzhen, China, November 5–9, 2019, Proceedings 21*, 353–369 (Springer, 2019).

10. Beneš, N., Brim, L., Huvar, O., Pastva, S. & Šafránek, D. Boolean network sketches: a unifying framework for logical model inference. *Bioinformatics* **39**, btad158 (2023).

11. Chevalier, S., Noël, V., Calzone, L., Zinovyev, A. & Paulevé, L. Synthesis and simulation of ensembles of boolean networks for cell fate decision. In *Computational Methods in Systems Biology: 18th International Conference, CMSB 2020, Konstanz, Germany, September 23–25, 2020, Proceedings 18*, 193–209 (Springer, 2020).

12. Helikar, T. *et al.* The cell collective: toward an open and collaborative approach to systems biology. *BMC systems biology* **6**, 1–14 (2012).

13. Malik-Sheriff, R. S. *et al.* BioModels—15 years of sharing computational models in life science. *Nucleic acids research* **48**, D407–D415 (2020).

14. Naldi, A. *et al.* Logical modeling and analysis of cellular regulatory networks with ginsim 3.0. *Front. physiology* **9**, 646 (2018).

15. Klarner, H., Streck, A. & Siebert, H. PyBoolNet: a python package for the generation, analysis and visualization of boolean networks. *Bioinformatics* **33**, 770–772 (2017).

16. Beneš, N., Brim, L., Kadlecaj, J., Pastva, S. & Šafránek, D. AEON: attractor bifurcation analysis of parametrised boolean networks. In *International Conference on Computer Aided Verification*, 569–581 (Springer, 2020).

17. Chaouiya, C. *et al.* SBML qualitative models: a model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools. *BMC systems biology* **7**, 1–15 (2013).

18. Van Ham, P. How to deal with variables with more than two levels. In *Kinetic Logic A Boolean Approach to the Analysis of Complex Regulatory Systems*, 326–343 (Springer Berlin Heidelberg, 1979).

19. Naldi, A. BioLQM: a java toolkit for the manipulation and conversion of logical qualitative models of biological networks. *Front. physiology* **9**, 1605 (2018).

20. Ostaszewski, M. *et al.* Covid-19 disease map, building a computational repository of sars-cov-2 virus-host interaction mechanisms. *Sci. data* **7**, 1–4 (2020).

21. Beneš, N. *et al.* AEON. py: Python library for attractor analysis in asynchronous Boolean networks. *Bioinformatics* **38**, 4978–4980 (2022).

22. Aghamiri, S. S. *et al.* Automated inference of boolean models from molecular interaction maps using casq. *Bioinformatics* **36**, 4473–4482 (2020).

23. Naldi, A. *et al.* The colomoto interactive notebook: accessible and reproducible computational analyses for qualitative biological networks. *Front. physiology* **9**, 680 (2018).

24. Su, C. & Pang, J. Cabean: a software for the control of asynchronous boolean networks. *Bioinformatics* **37**, 879–881 (2021).

25. Stoll, G. *et al.* Maboss 2.0: an environment for stochastic boolean modeling. *Bioinformatics* **33**, 2226–2228 (2017).

26. Paulevé, L., Kolčák, J., Chatain, T. & Haar, S. Reconciling qualitative, abstract, and scalable modeling of biological networks. *Nat. communications* **11**, 4256 (2020).

27. Rozum, J. C., Deritei, D., Park, K. H., Gómez Tejeda Zañudo, J. & Albert, R. pystablemotifs: Python library for attractor identification and control in boolean networks. *Bioinformatics* **38**, 1465–1466 (2022).

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 101034413.

Competing interests

The authors declare that there are no financial or non-financial competing interests associated with this work.

Figures & Tables

Online-only Table 1. List of all BBM models, including model ID, name, list of sources, and the corresponding publication. Also contains basic metadata of each model, such as the variable and regulation count.

Online-only Table 2. Associates BBM models with database entries. Lists a relevant BBM model reference for each entry in one of the tracked databases. Includes the date at which the model was last updated (in case of versioned database entries).

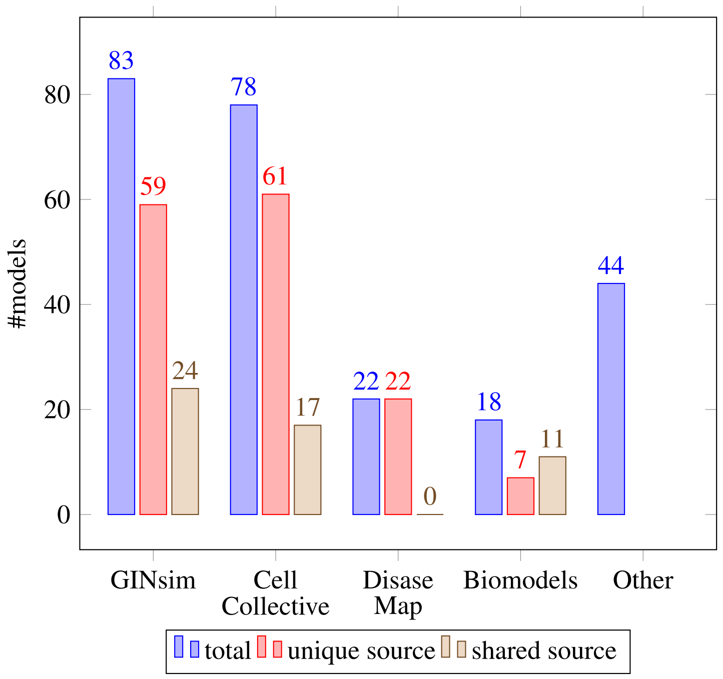

Figure 1. Models of the BBM dataset grouped by the database in which they appear, including the number of models uniquely available in each database and the models shared between multiple sources (see Online-only Table 2 for exact mapping).