Razor: Provenance and Exploration in Model-Finding [*]

Salman Saghafi and Daniel J. Dougherty

Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA 01609, USA

Abstract

Razor is a model-finder for first-order theories presented geometric form; geometric logic is a variant of first-order logic that focuses on "observable" properties. An important guiding principle of Razor is that it be accessible to users who are not necessarily expert in formal methods; application areas include software design, analysis of security protocols and policies, and configuration management.

A core functionality of the tool is that it supports *exploration* of the space of models of a given input theory, as well as presentation of *provenance* information about the elements and facts of a model. The crucial mathematical tool is the ordering relation on models determined by homomorphism, and Razor prefers models that are *minimal* with respect to this homomorphism-ordering.

1 Introduction

In this paper we report on some aspects of our model-finding tool Razor. Model-finding is a well-established technique in automated deduction, most typically used in hardware and software development and verification. Razor's goals are complementary to these uses of model-finding in formal verification: it is conceived as a tool to allow users to explore artifacts such as software designs, policies, protocols, and system configurations. Formally, this corresponds to exploring the space of models of a first-order specification. We specifically aim at users not expert in formal methods; in this regard our project is closer in spirit to Alloy [13] than to tools aimed primarily at verification. But our main focus is on *exploration.* We are interested in questions such as, "which models should the user see first?", "how can the user understand *why* this model satisfies the specification?" and "how can the user plot a path through space of all models of the specification?"

In previous work [18] the authors and colleagues have developed a tool, Aluminum, that was an initial exploration of these ideas, based on SAT-solving technology and evaluated specifically in the context of software engineering. Razor is an extension and generalization of Aluminum. The crucial differences are (i) a focus on the notion of homomorphism between models as a reflection of information content and (ii) specialization to theories presented in a convenient form: geometric logic [1, 5, 24]. These two aspects support exploration of individual models via provenance information for elements and facts, and exploration of the space of all models of a theory by augmentation. Crucial differences between Aluminum and Razor are that Aluminum does not treat languages with equality, and compares models with the same domain with respect to the property of being a submodel, which is a cruder notion of information-ordering.

There is natural preorder on models of a given signature, given by : $\mathbb{A} \preccurlyeq \mathbb{B}$ if there is a homomorphism from $\mathbb{A}$ to $\mathbb{B}$. We write $\mathbb{A} \approx \mathbb{B}$ if $\mathbb{A} \preccurlyeq \mathbb{B}$ and $\mathbb{B} \preccurlyeq \mathbb{A}$. This preordering has a natural "information content" interpretation: when $\mathbb{A} \preccurlyeq \mathbb{B}$, the model $\mathbb{B}$ has all the information in $\mathbb{A}$ (and perhaps more). Suppose a user is wondering whether a certain state of affairs S is consistent with a theory $\mathcal{T}$,

[*]This material is based upon work supported by the National Science Foundation under Grant No CNS-1116557

that is, she is looking for examples of $\mathcal{T}$ in which S holds. If $\mathbb{A}$ and $\mathbb{B}$ are models of $\mathcal{T} \cup \{S\}$, with $\mathbb{A} \preccurlyeq \mathbb{B}$, then exhibiting $\mathbb{A}$ conveys a clearer picture to the user: the extra information in $\mathbb{B}$ as compared with $\mathbb{A}$ is not required for S to hold, indeed this extra information is a distraction. We suggest that the ideal model to present to a user is one which contains only information *required to be present* in order to be a model of the underlying theory $\mathcal{T}$ and the state-of-affairs S in question. One only has to think of the familiar notion of a "minimal bug report" to appreciate the principle involved here. Of course there will not be *unique* minimal models for a theory, as there are for Horn theories. We understand "$\mathbb{M}$ is minimal" here to mean that for any model $\mathbb{N}$, if $\mathbb{N} \preccurlyeq \mathbb{M}$ then $\mathbb{M} \approx \mathbb{N}$.

We should note that Razor does not always return strictly minimal models: to do so is possible in theory but computationally expensive to guarantee (this is best explained after Razor's algorithm has been presented). The tool is intended as a lightweight formal method, allowing users to gain intuitions based on examples, and the role of minimality—typically obtained and never violated drastically—is to guide the choice of which models to display. See Example 8 for more discussion.

On the other hand, Razor is sound and refutationally complete (Theorem 4): models returned are guaranteed to be models of the input theory, and if the theory is unsatisfiable, Razor is guaranteed in principle to terminate with no models. Needless to say, if a theory has only infinite models, Razor will not terminate, and termination can be intractable for unsatisfiable theories.

The notion of *geometric theory* is defined in Section 2. As explained there, any first-order theory is equisatisfiable with a theory in geometric form and a wide class of specifications are actually logically equivalent with such theories. For theories in geometric form there is a model-finding approach, the *Chase,* that supports the construction of models with the following properties, which together comprise our notion of *provenance* in a model.

- *No junk, no confusion*:[1] Every element in the model is there for a reason: concretely, every element can be named by a Skolem term. Furthermore, if two names denote the same element in a model, there is a particular equation in the theory whose instantiation with those names induced this equality.

- *Justification*: Every fact in the model can be traced back to a specific instance of a particular axiom in the specification.

Put simply, a user can always ask, "why is this element in the model?", or "why is this fact true?", or "why these two elements equal?" and always be pointed to an axiom that can be "blamed".

Any one model by itself can only give limited information about the background theory. But it can be shown that all other models are obtained by adding facts to minimal ones. This suggests presenting the user with the capacity to play "what if?" games on a given model $\mathbb{M}$. Suppose a certain fact F does not hold in a model. The user can reasonably ask: "can I add F to the model $\mathbb{M}$?" The answer might be "no, F is inconsistent with $\mathcal{T}$". The answer might be "yes, but if so then G must also be true": that is, there may be other facts that follow logically from F in the presence of $\mathcal{T}$.

These two user-focused activities, exploring individual models via provenance information, and exploring the space of all models through augmentation, are the core functionalities of Razor.

We remark in passing that the ability to specify a theory using infinite disjunctions allows us to specify certain inductive relations, for example transitive closure, or adversary derivability in protocol analysis, going beyond first-order specification. We don't focus on that in this paper, though it requires very little change to our model-building apparatus: a fuller discussion of this phenomenon and its applications will appear elsewhere.

[1]We have not been able to resist this evocative terminology, typically applying to initial algebra specifications, even though our usage is subtly different from that setting.

The Name: We are inspired by Ockham's Razor, "Pluralitas non est ponenda sine neccesitate." Bertrand Russell [23] formulated the principle as follows: "Whenever possible, substitute constructions out of known entities for inferences to unknown entities."

Related Work

Model-Finding. The development of algorithms for the generation of relational models is an active area of research. The two most prominent methods are "MACE-style" [17], which reduce the problem to be solved into propositional logic and employ a SAT-solver, and "SEM-style" [25], which work directly in first-order logic. The goals of these works are mostly orthogonal to ours, since their concern is usually not the *exploration* of the space of all models of a theory. Lightweight formal methods tools such as Alloy [13] and Margrave [11, 19] also support model-finding as their primary activity.

Minimality. Logic programming languages produce single, *least*, models as a consequence of their semantics. Because user specifications are not limited to the Horn-clause fragment, our previous model-finding tool Aluminum can offer no such guarantee. The more general notion of minimal model in this paper has already been used in specifying the semantics of disjunctive logic programming [15] and of database updates [10] and in non-monotonic reasoning, especially circumscription [21].

In more specialized settings, generation of minimal models specifically usually relies on dedicated techniques, often based on tableaux (e.g., [20]) or hyperresolution (e.g., [4]). Aluminum [18] supports exploration by returning minimal models: it instruments the model-finding engine of Alloy. It thus inherits the limitation that it requires user-supplied bounds, and it cannot generate provenance information. The goals of the Cryptographic Protocol Shapes Analyzer [8] are closely aligned with ours; in analyzing protocols, it also generates minimal models. However, its application domain and especially algorithms are quite different from ours.

Geometric Logic. The case for geometric logic as a logic of observable properties was made clearly by Abramsky [1] and has been explored as a notion of specification by several authors [24], [12]. Geometric logic for theorem-proving was introduced in [3]. In [5] deNivelle generalized the setting of [3] to incorporate equality, and introduced a technique to augment the underlying theory as the system "learns" lemmas. The crucial difference with the current work is of course the fact that we focus on model-finding and exploration.

Chase. Our model-building is founded on the Chase, an algorithm well-known in the database community [2, 7, 16] Challenges arise for us in managing the complexity that arises due to disjunction, and in treating equality. Our strategies for addressing these challenges comprise much of the rest of this paper.

2 Geometric Logic and The Chase

We work over a first-order signature with relations symbols only. As syntactic sugar for users, we allow function symbols in the concrete syntax: formulas are translated in the usual way to eliminate function symbols in favor of relation symbols. Theories are augmented with axioms ensuring singled-validness but not necessarily with totality axioms. This means that we treat functions as being *partially defined.*

2.1 Geometric Logic

A *positive-existential* formula is one built over $\wedge$, $\exists$ and possibly-infinitary $\bigvee$. A *geometric sequent* is a construct $\varphi \vdash_{\vec{x}} \psi$, here φ and ψ are positive-existential formulas that contain free variables from a set of variables $\vec{x}$: this behaves semantically as $\forall \vec{x}.(\varphi \to \psi)$. We refer to φ and ψ respectively as *body* and *head* of $\varphi \vdash_{\vec{x}} \psi$. A *geometric theory* is a set of geometric sequents. (The term "geometric" arises from the original study of this class of formulas in the nexus between algebraic geometry and logic.)

If $\alpha(\vec{x})$ is a positive-existential formula true of a tuple $\vec{a}$ in a model $\mathbb{M}$ then the truth of this fact is witnessed by a finite fragment of $\mathbb{M}$. Thus if $\mathbb{M}$ satisfies α with $\vec{a}$ and is expanded, by adding new elements and/or new facts, $\alpha(\vec{x})$ still holds of $\vec{a}$ in the resulting model. For this reason, properties defined by positive-existential formulas are sometimes called *observable* properties [1]. More generally, positive-existential formulas are precisely the formulas preserved under homomorphisms; indeed this holds even if we restrict attention to finite models only [22].

By standard logical manipulations, we may assume that every geometric sequent $\varphi \vdash_{\vec{x}} \psi$ is in a simplified form

$$\alpha(\vec{x}) \vdash_{\vec{x}} \bigvee_j (\exists y_{j1} \ldots \exists y_{jp}. \beta_j(\vec{x}, y_{j1}, \ldots, y_{jp})),$$

where α and each β_j is a conjunction of atoms (perhaps equations). In the rest of this paper, we will assume geometric sequents to be in this form unless stated otherwise. Note that an empty conjunction may be regarded as truth ($\top$) and an empty disjunction head as falsehood ($\bot$). In this way a simple positive-existential formula, or the negation of such, comprises a geometric sequent itself.

Homomorphism A homomorphism between models $\mathbb{M}$ and $\mathbb{N}$ is a map from the domain of $\mathbb{M}$ to the domain of $\mathbb{N}$, $h : |\mathbb{M}| \to |\mathbb{N}|$, such that for every relational symbol R of arity n and any n-tuple $(\mathbf{e_1}, \ldots \mathbf{e_n})$ over the elements of $\mathbb{M}$, the fact $R(\mathbf{e_1}, \ldots, \mathbf{e_n}) \in \mathbb{M}$ implies $R(h(\mathbf{e_1}), \ldots, h(\mathbf{e_n})) \in \mathbb{N}$. We write $\mathbb{M} \preccurlyeq \mathbb{N}$ for the preorder defined by the existence of a homomorphism from $\mathbb{M}$ to $\mathbb{N}$, and $\mathbb{M} \approx \mathbb{N}$ for the reflexive closure of $\preccurlyeq$. We say that $\mathbb{M}$ is a *minimal model* in a class $\mathcal{C}$ of models if $\mathbb{N} \in \mathcal{C}$ and $\mathbb{N} \preccurlyeq \mathbb{M}$ implies $\mathbb{N} \approx \mathbb{M}$. A set $\mathcal{M}$ of models is a *set-of-support* for a class $\mathcal{C}$ if $\mathbb{N} \in \mathcal{C}$ implies that for some $\mathbb{M} \in \mathcal{M}$ we have $\mathbb{M} \preccurlyeq \mathbb{N}$.

Thus the homomorphism preorder captures the observable properties of models. In particular, if $\mathbb{M}$ is a minimal model for the class of models of geometric theory $\mathcal{T}$, then no elements or facts of $\mathbb{M}$ can be removed and still yield a model of T. This generalizes the well-known notion of minimal model for a Horn theory: the difference is that minimal models for a geometric theory need not be unique.

Skolemization Skolem functions will play an important role in the following, especially as regards provenance of elements. Given a sequent $\varphi \equiv \alpha \vdash_{\vec{x}} \bigvee_j (\exists y_{j1} \ldots \exists y_{jp}. \beta_j(\vec{x}, y_{j1}, \ldots, y_{jp}))$, we assign a unique, fresh, *Skolem function symbol* $\mathsf{f}^{\sigma}_{\mathsf{jk}}$ to every existential quantifier $\exists y_{jk}$, deriving an associated, Skolemized, sequent $\alpha \vdash_{\vec{x}} \bigvee_j \beta_j(\vec{x}, f^{\sigma}_{j1}(\vec{x}), \ldots, f^{\sigma}_{jp}(\vec{x}))$.

Notation 1. *To make it convenient for the reader to refer back and forth between an existential quantifier and the associated Skolem function, we will adopt the following convention:* a quantifier that induces the Skolem function f may be decorated with f as a superscript in the text. *This decoration has no logical significance for the formula itself, it is simply a convenient meta-notation recording the association. See Example 13, for instance.*

We stress that this Skolemization is not a source-transformation of the input theory, it is a component of the under-the-hood activity of the Razor tool. The models that Razor constructs are models of the user's original input language. The reasons for this distinction are subtle, but important.

Skolemization is sound for refutation theorem-proving since a formula is equisatisfiable with its Skolemization. The situation for model-finding is different. We do not view Skolemization as expanding the signature for our models, since the notion of homomorphism is central for us, and expanding the signature changes the notion of homomorphism: specifically, we do not want to insist that homomorphisms preserve Skolem functions. Rather, we view Skolemization as the passage from a sequent σ to a new sequent σ' that is *existentially quantified* by the new function symbols. Note that in this second-order reading, σ' is actually logically equivalent to the original σ. Models constructed by the Chase have the property that every element is added to the model to satisfy an (original) existential quantifier, so, in a very real sense, the process of model-finding is nothing more than the incremental construction of witnesses for these Skolem functions. The expanded signature with Skolemization provides a language to communicate provenance to the user, and also plays a role in bounded model-finding (Section 3.4).

Expressiveness Any first-order theory is equisatisfiable to a geometric theory, via Skolemization and passage to conjunctive normal form. So geometric logic does not offer itself as a tractable class for model-finding, since satisfiability is undecidable. The virtue of geometric form is that it lends itself to a natural way of constructing models: the Chase.

2.2 The Chase

The standard Chase algorithm proceeds as follows. When the input theory contains disjunctions (in the consequents of sequents) the Chase must branch in order to do an exhaustive search for minimal models. It will be easiest to present the algorithm as a non-deterministic procedure. Various runs of the Chase induced by the non-deterministic choices lead to various models of the theory. Let C be an infinite set of fresh constants. A *fact* over C is a closed atomic sentence over C. Now suppose $\mathbb{M}$ is a set of facts over C viewed as candidate model for the theory. If $\mathbb{M}$ is not yet a model of $\mathcal{T}$ then there is some sequent

$$\sigma \equiv \alpha(\vec{x}) \vdash_{\vec{x}} \bigvee_j \gamma_j(\vec{x})$$

and environment $\eta \equiv \{x_1 \mapsto c_1, \ldots, x_k \mapsto c_k\}$ making the sequent false in $\mathbb{M}$, *i.e.*, $\alpha(\eta\vec{x})$ holds in $\mathbb{M}$ yet for no j do we have $\gamma_j(\eta\vec{x})$ true in $\mathbb{M}$. A chase-step on $\mathbb{M}$, σ, and η that returns a new candidate model $\mathbb{M}'$ is denoted by $\mathbb{M} \xrightarrow{(\sigma,\eta)} \mathbb{M}'$ and proceeds as follows:

Operation 1 choose some disjunct $\gamma_j \equiv \exists z_{j1} \ldots z_{jp} \,.\, R_{j1} \wedge \ldots \wedge R_{jn}$;

Operation 2 add new elements $\mathbf{e_1}, \ldots, \mathbf{e_p}$ to C ;

Operation 3 add each of the facts $\eta' R_{ji}$ $(1 \le i \le n)$ to $\mathbb{M}$, where η' is the environment obtained by adding the bindings $z_{jk} \mapsto \mathbf{e_k}$ $(1 \le k \le p)$ to η. If any of the R_{ji} is an equality, then we induce an identification between the appropriate elements of $\mathbb{M}$.

The *Chase* consists of starting with the empty set of facts and iterating chase-steps. We halt with success if we reach a set $\mathbb{M}$ of facts where we cannot apply a step, *i.e.* when $\mathbb{M}$ is a model of $\mathcal{T}$. We halt with failure if we reach a set $\mathbb{M}$ of facts where a sequent with empty head fails in $\mathbb{M}$ (which is to say, its body is true): we cannot "repair" $\mathbb{M}$ to make such a sequent true.

We now prove a lemma that will be later used to prove our main theorem about the Chase:

Lemma 2. *Let $\mathcal{T}$ be a geometric theory and $\mathbb{M}$ a model of $\mathcal{T}$. Let $\mathbb{N}$ be a structure such that $\mathbb{N} \preccurlyeq \mathbb{M}$. Then for an environment η and sequent $\sigma \in \mathcal{T}$, if $\mathbb{N}$ is not a model of $\mathcal{T}$ in η, there exists a chase-step $\mathbb{N} \xrightarrow{(\sigma,\eta)} \mathbb{N}'$ where $\mathbb{N}' \preccurlyeq \mathbb{M}$.*

Proof. Let σ be $\varphi \vdash_{\vec{x}} \psi$. Because $\mathbb{N}$ is not a model of σ, φ is true under η in $\mathbb{N}$ but ψ is false under η. Letting h be the homomorphism from $\mathbb{M}$ to $\mathbb{N}$, φ is true in $\mathbb{M}$ under $(h \circ \eta)$. Since $\mathbb{M}$ is a model of $\mathcal{T}$, some disjunct $\exists y_1 \ldots \exists y_p . \beta(\vec{x}, y_1, \ldots, y_p))$, of ψ must also be true in $\mathbb{M}$ under $(h \circ \eta)$. That is, there are elements $\mathbf{d_1}, \ldots \mathbf{d_p}$ of $\mathbb{M}$ with β true in $\mathbb{M}$ under $(h \circ \eta^+)$ where η^+ extends η by mapping the y_i to the d_i.

There exists a chase-step $\mathbb{N} \xrightarrow{(\sigma,\eta)} \mathbb{N}'$ that chooses this disjunct: it extends $\mathbb{N}$ by adding fresh elements $\mathbf{e_1},\ldots,\mathbf{e_p}$ if $p > 0$, and ensures that β is true in $\mathbb{N}'$ under η', where η' extends η by mapping the y_i to the e_i.

If we extend h to map each e_i to d_i, $1 \leq i \leq p$, it is easy to check that the resulting map is a homomorphism from $\mathbb{N}'$ to $\mathbb{M}$. □

2.2.1 Fairness

A deterministic implementation of the standard Chase that utilizes a *scheduler* to select the sequent σ and the environment η is said to be *fair* if the scheduler guarantees that every pair of possible choices for σ and η will be eventually evaluated. Definition 3 captures this idea formally:

Definition 3. Let $\mathcal{T}$ be a geometric theory and ρ be an infinite run of the chase starting from an empty model $\mathbb{M}_0$:

$$\rho = \mathbb{M}_0 \xrightarrow{(\sigma_0,\eta_0)} \mathbb{M}_1 \xrightarrow{(\sigma_1,\eta_1)} \ldots \mathbb{M}_i \xrightarrow{(\sigma_i,\eta_i)} \mathbb{M}_{i+1} \xrightarrow{(\sigma_{i+1},\eta_{i+1})} \ldots$$

Let $\mathcal{D} = \bigcup\{|\mathbb{M}_k| \mid 0 \leq k\}$ be a domain of elements, where $|\mathbb{M}_k|$ denotes the domain of $\mathbb{M}_k$. We say that ρ is a fair run of the Chase if for every (σ,η), where $\sigma \in \mathcal{T}$ and η is an environment from the free variables of σ to the elements of $\mathcal{D}$, and a structure $\mathbb{M}_j$ such that $\mathbb{M}_j$ is not a model of σ in environment η, there exists i such that $j \leq i$ and $(\sigma_i,\eta_i) = (\sigma,\eta)$.

The following records the basic properties of the Chase; each result is either well-known or a routine generalization of known facts. Note that it is possible that the Chase may not halt. In this case, if the Chase is done in a fair manner, the resulting infinite set of facts will be a model of $\mathcal{T}$.

Theorem 4. *Let $\mathcal{T}$ be a geometric theory. Then $\mathcal{T}$ is satisfiable if and only if there is a fair run of the Chase which does not fail. Let $\mathcal{U}$ be the set of models obtained by some execution of the Chase (here we include the possibility of infinite runs). For any model $\mathbb{M}$ of $\mathcal{T}$, there is a $\mathbb{U} \in \mathcal{U}$ and a homomorphism from $\mathbb{U}$ to $\mathbb{M}$.*

Proof. We show that for every model $\mathbb{N}$ of $\mathcal{T}$, there exists a run ρ of the Chase computing a model $\mathbb{M}$ where $\mathbb{M} \preccurlyeq \mathbb{N}$: Let $\mathbb{M}_0$ be the empty model. The empty function $h : |\mathbb{M}_0| \to |\mathbb{N}|$ is a trivial homomorphism, thus, $\mathbb{M}_0 \preccurlyeq \mathbb{N}$. Create a fair run ρ by iterating Lemma 2 on the models generated by successive chase-steps starting from $\mathbb{M}_0$:

$$\rho = \mathbb{M}_0 \xrightarrow{(\sigma_0,\eta_0)} \mathbb{M}_1 \xrightarrow{(\sigma_1,\eta_1)} \ldots \mathbb{M}_i \xrightarrow{(\sigma_i,\eta_i)} \mathbb{M}_{i+1} \xrightarrow{(\sigma_{i+1},\eta_{i+1})} \ldots$$

If the Chase stops after n steps with $\mathbb{M}_n$, then $\mathbb{M}_n$ is a model of $\mathcal{T}$, and $\mathbb{M}_n \preccurlyeq \mathbb{N}$ by Lemma 2. Conversely, if the Chase never stops, the resulting structure $\mathbb{M}_\infty$ will be infinite. Lemma 2 yields $\mathbb{M}_\infty \preccurlyeq \mathbb{N}$, thus, it suffices to show $\mathbb{M}_\infty$ is a model of $\mathcal{T}$. That is, for every sequent $\sigma \equiv \varphi \vdash_{\vec{x}} \psi$ in $\mathcal{T}$ and every environment $\eta : \vec{x} \to |\mathbb{M}_\infty|$, $\mathbb{M}_\infty$ is a model of σ in η: clearly, if φ is not true in $\mathbb{M}_\infty$, σ is true in $\mathbb{M}_\infty$. Otherwise, if φ is true in $\mathbb{M}_\infty$, then for some i, $\eta\vec{x} \subseteq |\mathbb{M}_i|$. Since ρ is fair, there exists a chase-step $\mathbb{M}_j \xrightarrow{(\sigma_{j+1},\eta_{j+1})} \mathbb{M}_{j+1}$ for some j where $i \leq j$. Finally, because $\mathbb{M}_{j+1}$ is a submodel of $\mathbb{M}_\infty$ and ψ is positive, ψ is true in $\mathbb{M}_\infty$ for environment η, hence σ holds. □

The set $\mathcal{U}$ in Theorem 4 is said to be a set of jointly universal models for $\mathcal{T}$.

A subtle point is that simply running the Chase does not ensure that the models returned are strictly minimal (Example 8): certain anomalies may cause a naive chase sequence to return a non-minimal model (though these are still models of the input theory, and it is guaranteed that every minimal model will certainly be obtained). In theory this can be addressed by post-processing of the space of models generated, but this is expensive and non-minimality seems not arise often in practice, so the default is to return models as computed by the Chase.

2.2.2 Termination

In general, termination of the Chase for an arbitrary geometric theory is undecidable [6]. However, Fagin *et al.* [9] define a syntactic condition on geometric theories, known as *weak acyclicity*, by which the Chase is guaranteed to terminate. Briefly, one constructs a directed graph whose nodes are positions in relations and whose edges capture possible "information flow" between positions; a theory is weakly acyclic if there are no cycles of a certain form in this graph. (The notion of weakly acyclicity in [9] is defined for theories without disjunction, but the obvious extension of the definition to full geometric logic supports the argument for termination in the general case.)

Observe that if $\mathcal{T}$ is such that all runs of the Chase terminate, then—by König's Lemma—there is a finite set of models returned by the Chase. Thus we can compute a finite set that jointly provides a set-of-support for all models of $\mathcal{T}$ relative to the homomorphism order $\preccurlyeq$.

For theories that do not guarantee termination of the Chase, we must resort to bounding our search arbitrarily. A simple way to do so, used by Alloy, Margrave, and Aluminum, is to use user-supplied upper bounds on the domain of the model. In Section 3.3, we describe a somewhat more subtle device in Razor, based on Skolemization, which allows for *controlled* confusion in order to bound the search.

2.2.3 Chase Examples

Soon we introduce an extended running example for the paper but we pause here for some simple examples to demonstrate the Chase.

Example 5.

$$\begin{aligned} &\vdash \exists x\, y\, .\, R(x,y)\\ R(x,w) &\vdash \exists y\, .\, Q(x,y)\\ Q(u,v) &\vdash \exists z\, .\, R(u,z) \end{aligned}$$

Starting with an empty model, in the first step we treat the first sequent: we create new element $\mathbf{e_1}$ and $\mathbf{e_2}$ and add the fact $R(\mathbf{e_1},\mathbf{e_2})$. Then the second sequent fails, so we add an element $\mathbf{e_3}$ to instantiate $\exists y$, and add the fact $Q(\mathbf{e_1},\mathbf{e_3})$. The third sequent is satisfied in the model $\{R(\mathbf{e_1},\mathbf{e_2}), Q(\mathbf{e_1},\mathbf{e_3})\}$ so the procedure terminates with this model.

Example 6. This example is a very slight syntactic modification of the previous one (in the head of the third sequent).

$$\begin{aligned} &\vdash \exists x\, y\, .\, R(x,y)\\ R(x,w) &\vdash \exists y\, .\, Q(x,y)\\ Q(u,v) &\vdash \exists z\, .\, R(z,v) \end{aligned}$$

In this example, the first two steps are the same as above, yielding $\{R(\mathbf{e_1},\mathbf{e_2}), Q(\mathbf{e_1},\mathbf{e_3})\}$. But now the third sequent requires creation of a new element $\mathbf{e_4}$ and the fact $R(\mathbf{e_4},\mathbf{e_3})$. Then the second sequent fails in the model $\{R(\mathbf{e_1},\mathbf{e_2}), Q(\mathbf{e_1},\mathbf{e_3}), R(\mathbf{e_4},\mathbf{e_3})\}$ and so we add another element ... and it is easy to see that the Chase will never terminate on this theory.

Note, however, that the infinite run of the Chase that is induced does create an (infinite) model of the theory. As noted above, this is a general fact about fair Chase runs. Although such infinite models are not presented as output of the tool, the fact that the process creates models "in the limit" is a necessary component of the completeness proof: *unsatisfiable* theories will necessarily lead to finitely-failing computations.

Example 7. This example demonstrates the importance of fairness for correctness of Theorem 4:

$$\begin{aligned} &\vdash \exists x\, y\, .\, R(x,y)\\ R(x,y) &\vdash \exists z\, .\, R(y,z)\\ R(x,y) &\vdash \bot \end{aligned}$$

Consider an "unfair" run of the Chase, which starts with the first sequent, then, continuously favors the second sequent over the last one: it repeatedly creates new facts $R(\mathbf{e_1},\mathbf{e_2})$, $R(\mathbf{e_2},\mathbf{e_3})$, $R(\mathbf{e_3},\mathbf{e_4})$, $\ldots$, thus, never terminates. However, a fair run of the Chase will eventually treat the last sequent and terminate with failure.

Example 8 (A Failure of Minimality)**.** To gain some intuition about minimality and disjunction, consider the (propositional) theory

$$\begin{aligned} &\vdash (A \vee B)\\ A &\vdash B \end{aligned}$$

The Chase can generate two models here, one of which fails to be minimal. What has gone wrong is that the disjunction is spurious, in the sense that one disjunct is subsumed by the other in the context of the rest of the theory. This little example makes it clear that any implementation of the Chase that explores disjuncts without doing global checks for the relationships among the models being constructed can end up creating models that are not minimal, essentially because the theory the user presented is not as "tight" as it might be.

This is an interesting phenomenon, since it is still the case that, even in a non-minimal model, no element or fact is created except when it is required: any model constructed by Razor will have complete provenance information. But since homomorphisms are defined with respect to the original (user's) input language, homomorphisms need not respect provenance information. This opens the door to failures of minimality such as shown above.

It is also clear that such non-minimality could be avoided at the cost of post-processing the output. We do not do this, simply because achieving such absolute minimality does not seem to be worth the computational overhead.

2.3 Extended Example: A File System

We use a translation to geometric logic of the filesystem specification from Alloy's distribution (at `http://alloy.mit.edu/download.html`) as a running example throughout this paper (Figure 1).

Capturing Type Information: The first six sequents in Figure 1 can be viewed as capturing type information. For instance, files and directories are disjoint subtypes of filesystem objects (Sequents 1, 2, 3 and 4); *Live* is a relation between a filesystem and a filesystem object (Sequent 5); and, *root* is a function that takes a filesystem as input and returns a directory as output (Sequent 6). Sequent 14 forbids recursive contents in a filesystem.

$$
\begin{array}{rcll}
FSObject(o) & \vdash & Dir(o) \vee File(o) & (1)\\
File(f) & \vdash & FSObject(f) & (2)\\
Dir(d) & \vdash & FSObject(d) & (3)\\
File(o) \wedge Dir(o) & \vdash & \bot & (4)\\
Live(fs,o) & \vdash & FileSystem(fs) \wedge FSObject(o) & (5)\\
root(fs) = r & \vdash & FileSystem(fs) \wedge Dir(r) & (6)\\
 & \dots & & \\
parent(fs,o) = p & \vdash & Live(fs,o) \wedge Live(fs,p) \wedge Dir(p) & (7)\\
Live(fs,o) & \vdash & (\exists^{\mathsf{hasParent}} p.\ parent(fs,o) = p) & (8)\\
 & & \qquad \vee (root(fs) = o) & (9)\\
(root(fs) = r) \wedge (parent(fs,r) = p) & \vdash & \bot & (10)\\
parent(fs,o) = p & \vdash & Contents(fs,p,o) & (11)\\
Contents(fs,d,o) & \vdash & Contents^*(fs,d,o) & (12)\\
Contents(fs,d_1,d_2) \wedge Contents^*(fs,d_2,o) & \vdash & Contents^*(fs,d_1,o) & (13)\\
Contents^*(fs,o_1,o_2) \wedge Contents^*(fs,o_2,o_1) & \vdash & \bot & (14)\\
\top & \vdash & \exists^{\mathsf{someFileSys}} fs.\ \exists^{\mathsf{someObject}} o. & (15)\\
 & & \quad FileSystem(fs) \wedge Live(fs,o) & \\
 & \dots & &
\end{array}
$$

Figure 1: Filesystem Example [2]

Some Axioms: According to Sequent 7, every object and its parent in a filesystem must live in the filesystem, and the parent must be a directory. Sequent 8 expresses that every object in a filesystem must have a parent unless it is the root. The Skolem function hasParent has been assigned to the existential quantifier in the head of this sequent, which is used for constructing provenance information. Sequent 10 does not allow the root of a filesystem to have a parent in the filesystem. Sequent 11 states that every object in a filesystem is a content of its parent. Finally, because of Sequent 15, every model of the theory has to contain at least a filesystem and an object that lives in that filesystem. The Skolem functions someFileSys and someObject have been assigned to the existential quantifiers in the head of this sequent.

Transitive Closure: The axioms corresponding to the transitive closure of $Contents$ are captured by Sequents 12 and 13, serving as an inductive definition for $Contents^*$. Notice that because the Chase only adds $Contents^*$ facts when required by the theory, the relation $Contents^*$ is guaranteed to be the least transitive relation over $Contents$.

[2]By the convention established in Notation 1: the existential quantifiers of the theory are tagged with Skolem functions, namely hasParent, someFileSys and someObject

3 Running Razor

Razor's mode of interaction is a read-eval-print loop. Typical user commands include

- load a geometric theory, implicitly generating a stream of models
- navigate through the stream of models, via next and back commands
- augment the current model by adding elements or facts, generating a new stream of models witnessing the augmentation plus anything entailed in the context of the current theory.
- asking for the provenance, or justification, of an element or a fact.

3.1 Constructing Models

The fundamental operation of Razor is the use of the Chase to construct models. The Chase was described in the previous section, and some of the implementation design choices are described in Section 4.

3.1.1 Filesystem Example: Constructing Models

Note that the empty structure is a model of any sequent that has a non-trivial body. So any Razor computation starts with treating those sequents with empty bodies, such as sequent 15, in the filesystem example.

Because of Sequent 15, every run of the Chase for the theory adds two fresh elements $\mathbf{e_1}$ and $\mathbf{e_2}$ to the initial empty model and makes $FileSystem(\mathbf{e_1})$ and $Live(\mathbf{e_1}, \mathbf{e_2})$ true in it.

Next, because the body of Sequent 8 is true but its head is false in the environment $\{fs \mapsto \mathbf{e_1}, o \mapsto \mathbf{e_2}\}$, the Chase will make a chase-step on Sequent 8. This may be done by making either of the two disjuncts in the head true:

- Corresponding to the second disjunct of the head, the fact $root(\mathbf{e_1}) = \mathbf{e_2}$ may be added to the model.

  After this, the body of 6 will be true but its head will be false for $\{fs \mapsto \mathbf{e_1}, r \mapsto \mathbf{e_2}\}$; thus, $Dir(\mathbf{e_2})$ will be added to the model. Eventually, after another chase-step on Sequent 3 in environment $\{d \mapsto \mathbf{e_2}\}$, the resulting model will satisfy the theory:

  $$\mathbb{F}_1 = \{FileSystem(\mathbf{e_1}), Live(\mathbf{e_1}, \mathbf{e_2}), root(\mathbf{e_1}) = \mathbf{e_2}, Dir(\mathbf{e_2}), FSObject(\mathbf{e_2})\}$$

- Alternatively, corresponding to the first disjunct of the head, a fresh element $\mathbf{e_3}$ and the fact $parent(\mathbf{e_1}, \mathbf{e_2}) = \mathbf{e_3}$ may be added to the model. Consequently, after processing Sequent 7 and Sequent 11 for $\{fs \mapsto \mathbf{e_1}, o \mapsto \mathbf{e_2}, p \mapsto \mathbf{e_3}\}$, Sequent 12 for $\{fs \mapsto \mathbf{e_1}, o \mapsto \mathbf{e_2}, d \mapsto \mathbf{e_3}\}$, and adding the corresponding facts to the model, once again, the body of Sequent 8 will become true for $\{fs \mapsto \mathbf{e_1}, o \mapsto \mathbf{e_3}\}$ while its head is false. Again, the Chase can make the head of Sequent 8 true by choosing either of the two disjuncts in the head.

If the Chase chooses to make $root(fs) = \mathbf{e_3}$, the next two models will be returned:

$$\begin{aligned}
\mathbb{F}_2 = \{&FileSystem(\mathbf{e_1}), Live(\mathbf{e_1},\mathbf{e_2}), parent(\mathbf{e_1},\mathbf{e2}) = \mathbf{e_3}, Contents(\mathbf{e_1},\mathbf{e_3},\mathbf{e_2}),\\
&Contents^*(\mathbf{e_1},\mathbf{e_3},\mathbf{e_2}), Live(\mathbf{e_1},\mathbf{e_3}), root(\mathbf{e_1}) = \mathbf{e_3},\\
&Dir(\mathbf{e_3}), FSObject(\mathbf{e_3}), File(\mathbf{e_2}), FSObject(\mathbf{e_2})\}\\
\mathbb{F}_3 = \{&FileSystem(\mathbf{e_1}), Live(\mathbf{e_1},\mathbf{e_2}), parent(\mathbf{e_1},\mathbf{e2}) = \mathbf{e_3}, Contents(\mathbf{e_1},\mathbf{e_3},\mathbf{e_2}),\\
&Contents^*(\mathbf{e_1},\mathbf{e_3},\mathbf{e_2}), Live(\mathbf{e_1},\mathbf{e_3}), root(\mathbf{e_1}) = \mathbf{e_3},\\
&Dir(\mathbf{e_3}), FSObject(\mathbf{e_3}), Dir(\mathbf{e_2}), FSObject(\mathbf{e_2})\}
\end{aligned}$$

Alternatively, if the Chase chooses the first disjunct of Sequent 8 and allows a fresh element $\mathbf{e_4}$ to be the parent of $\mathbf{e_3}$, the sequent will again have the opportunity to be processed under $\{fs \mapsto \mathbf{e_1}, o \mapsto \mathbf{e_4}\}$. This situation arises for every new element that is created to be the parent of an existing one: it is easy to see that a run of the Chase that always chooses the first disjunct of Sequent 8 corresponds to an infinite model and will not terminate. In addition, the theory has an infinite number of models as there is no bound on the depth of filesystem trees.

3.2 Model Augmentation

Let $\mathcal{T}$ be a geometric theory and $\mathbb{M}$ be a model of $\mathcal{T}$. Razor allows the user to *augment* $\mathbb{M}$ with an additional "observation" α resulting in an extension model $\mathbb{N}$ of $\mathcal{T}$ such that $\alpha \in \mathbb{N}$. The observation α can in principle be an arbitrary positive-existential formula, though clearly there is no benefit to the user in allowing disjunction. So in practice we accept any possibly-existentially quantified conjunction of atomic sentences referring to elements of the model.

Due to the monotonicity of our model-building process, $\mathbb{N}$ may simply be computed by a run of the Chase starting with a model $\mathbb{M}' \equiv \mathbb{M} \cup \{\alpha\}$. Such a run of the Chase starting from $\mathbb{M}'$ will fail if $\mathbb{M}$ is inconsistent with α according to $\mathcal{T}$. Similarly, if α entails other observations given $\mathcal{T}$ and the facts already in $\mathbb{M}$, those observations will be added to the resulting model.

3.2.1 Filesystem Example: Augmentation

Consider model $\mathbb{F}_1$ from the filesystem example. Assume the user is conjecturing that it is possible for the root of the filesystem in this model to be the parent of another object. The user can test her conjecture by augmenting $\mathbb{F}_1$ with the fact $parent(\mathbf{e_1},\mathbf{e_3}) = \mathbf{e_2}$ where $\mathbf{e_3}$ is a fresh element that currently does not live in $\mathbb{F}_1$.

Razor verifies the conjecture by a run of the Chase starting from $\mathbb{F}'_1 \equiv \mathbb{F}_1 \cup \{parent(\mathbf{e_1},\mathbf{e_3}) = \mathbf{e_2}\}$. This run will add $Live(\mathbf{e_1},\mathbf{e_3})$ to $\mathbb{F}'_1$ due to a chase-step on Sequent 7 and environment $\{fs \mapsto \mathbf{e_1}, o \mapsto \mathbf{e_3}, p \mapsto \mathbf{e_2}\}$. Consequently, after a few more steps on Sequents 5, 11, 12 and 1, the Chase will terminate with either of the two following models:

$$\begin{aligned}
&\mathbb{F}'_1 \cup \{Live(\mathbf{e_1},\mathbf{e_3}), FSObject(\mathbf{e_3}), File(\mathbf{e_3}), Contents(\mathbf{e_1},\mathbf{e_2},\mathbf{e_3}), Contents^*(\mathbf{e_1},\mathbf{e_2},\mathbf{e_3})\}\\
\text{or}\quad &\mathbb{F}'_1 \cup \{Live(\mathbf{e_1},\mathbf{e_3}), FSObject(\mathbf{e_3}), Dir(\mathbf{e_3}), Contents(\mathbf{e_1},\mathbf{e_2},\mathbf{e_3}), Contents^*(\mathbf{e_1},\mathbf{e_2},\mathbf{e_3})\}
\end{aligned}$$

The result of this operation shows that the user's conjecture is true: the specification allows for the root of the filesystem to be the parent of a file or a directory, nevertheless, in presence of additional information implied by the new fact.

Alternatively, if the user attempts to augment a model with inconsistent facts, Razor will not return any models. This behavior often comes in handy when verifying invariants that must hold in the models of the theory. For example, if the user chooses to augment $\mathbb{F}_2$ from the running example with a new fact $root(\mathbf{e_1}) = \mathbf{e_2}$, no models will be returned: since element $\mathbf{e_3}$ is the parent of $\mathbf{e_2}$ in filesystem $\mathbf{e_1}$, because of Sequent 10 (see Figure 1), $\mathbf{e_2}$ cannot be the root of the filesystem.

3.3 Provenance Computation

Here we make precise the sense in which Razor computes provenance information. We first define the notions of *name* and *justification* for the elements and facts of models.

Definition 9. Consider a chase-step $\mathbb{M} \xrightarrow{(\sigma,\eta)} \mathbb{N}$ for sequent $\sigma \equiv \varphi \vdash_{\{x_1,\dots,x_m\}} \psi$. Let $\exists_1^{f_1} y_1 \dots \exists_p^{f_p} y_p.E$ be the disjunct in ψ selected by the algorithm (*Operation 1*), and let $\mathbf{e_1},\dots,\mathbf{e_p}$ be the set of fresh elements to be added to $\mathbb{M}$ (*Operation 2*). For each i, let $\mathsf{f_i}$ be the Skolem function associated with the ith existential quantifier. Then the *name* of element $\mathbf{e_i}$ in the new model $\mathbb{N}$, written as $\mathrm{N}_{\mathbb{N}}(\mathbf{e_i})$, is a closed term defined by the following:

$$\mathrm{N}_{\mathbb{N}}(\mathbf{e_i}) = \mathsf{f_i}(\mathrm{N}_{\mathbb{M}}(\eta(x_1)),\dots,\mathrm{N}_{\mathbb{M}}(\eta(x_m)))$$

Note that this term depends on the sequent in question, the model $\mathbb{M}$, and the environment η.

Definition 10. Let $\mathbb{M} \xrightarrow{(\sigma,\eta)} \mathbb{N}$ be a chase-step. Let $\mathcal{F}$ be the set of facts to be added to $\mathbb{N}$ (*Operation 3*). Then we say the pair (σ,η) is the *justification* of a $F \in \mathcal{F}$ in $\mathbb{N}$, denoted by $\Gamma_{\mathbb{N}}(F)$.

Names and justifications connect the elements and the facts of a model, which is constructed by some run of the Chase, to fragments of the theory that necessitate the existence of those elements and facts. Razor constructs the name and the justification data in separate computational contexts in parallel with a run of the Chase and presents the results of those computations as explanatory provenance information for models. The user can use this information to understand why a particular element exists in a model, why a fact is true, or why two elements are equal, bridging the gap between the model and the theory.

Theorem 11. *Let $\mathbb{M}$ be a model over the domain $|\mathbb{M}|$, which is constructed by Razor for a theory $\mathcal{T}$. Then*

- *every element $\mathbf{e} \in |\mathbb{M}|$ is named by a term $t \equiv \mathrm{N}_{\mathbb{M}}(\mathbf{e})$ over the Skolem functions that are associated to the existential quantifiers of $\mathcal{T}$.*
- *every fact $F \in \mathbb{M}$ is justified by a pair $(\sigma,\eta) \equiv \Gamma_{\mathbb{M}}(F)$ for $\sigma \in \mathcal{T}$ and $\eta : \vec{x} \to |\mathbb{M}|$, where $\vec{x}$ is the set of free variables in σ.*

Theorem 11 is trivially true because every element and fact in a model that Razor constructs is added as the result of some chase-step. This automatically assigns a name to every element and a justification to every fact.

Informally speaking, every piece of information, whether an element or a fact, in a model constructed by Razor is necessary, that is, it is the product of some chase-step, necessary for repairing a sequent of the input theory in the model.

Razor, however, allows for *controlled* confusion in order to bound the search for models:

Definition 12. Fix a natural number d. We say that two ground terms s and t are *d-equivalent,* written $s \approx_d t$, if they agree, as trees, up to depth d.

Razor constructs the provenance and justification information as byproducts of a run of the Chase, in parallel computational contexts.

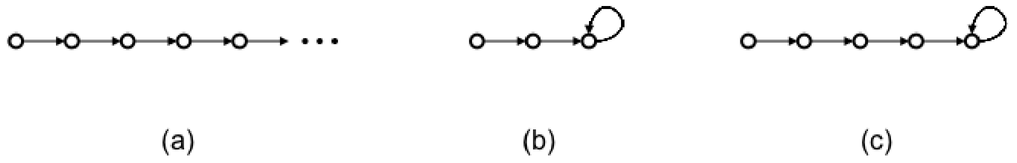

Figure 2: Bounding by Skolem Depth

3.3.1 Filesystem Example: Provenance

Recall model $\mathbb{F}_2$ for the filesystem example. Suppose that the user is interested in an explanation for element $\mathbf{e_3}$ in $\mathbb{F}_2$. The provenance of $\mathbf{e_3}$ in $\mathbb{F}_2$ is computed as follows:

1. Notice that Sequent 15 in Figure 1 contains no free variables, thus, Skolem functions $\mathsf{someFileSys}$ and $\mathsf{someObject}$ are nullary. Therefore, when applying a chase-step on Sequent 15, the names of the freshly generated elements $\mathbf{e_1}$ and $\mathbf{e_2}$ are respectively computed as $\mathrm{N}_{\mathbb{F}_2}(\mathbf{e_1}) = \mathsf{someFileSys}$ and $\mathrm{N}_{\mathbb{F}_2}(\mathbf{e_2}) = \mathsf{someObject}$.
2. When making a chase-step on Sequent 8 in environment $\{fs \mapsto \mathbf{e_1}, o \mapsto \mathbf{e_2}\}$, yielding element $\mathbf{e_3}$, the name $\mathrm{N}_{\mathbb{F}_2}(\mathbf{e_3}) = \mathsf{hasParent}(\mathsf{someFileSys}, \mathsf{someObject})$ is computed.

Intuitively, the term $\mathsf{hasParent}(\mathsf{someFileSys}, \mathsf{someObject})$ provides the user with the following explanation for $\mathbf{e_3}$: the (constant) terms $\mathsf{someFileSys}$ and $\mathsf{someObject}$ imply the existence of two elements (*i.e.*, $\mathbf{e_1}$ and $\mathbf{e_2}$) in $\mathbb{F}_2$. Besides, the theory necessitates that the object named by the latter *must* have a parent in the filesystem named by the former: element $\mathbf{e_3}$ is the parent.

The user may also ask "why $\mathbf{e_3}$ is a directory in $\mathbb{F}_2$?". It is easy to see that the justification information for $Dir(\mathbf{e_3})$, $\Gamma_{\mathbb{F}_2}(Dir(\mathbf{e_3}))$, is the pair $(Sequent\ 7, \{fs \mapsto \mathbf{e_1}, o \mapsto \mathbf{e_2}, p \mapsto \mathbf{e_3}\})$: intuitively, because $\mathbf{e_3}$ is the parent of $\mathbf{e_2}$ in filesystem $\mathbf{e_1}$, Sequent 7 requires it to be a directory.

3.4 Bounded Model-Finding

There is a natural way to view the set of closed terms as a metric space, defining the distance between distinct terms s and t is 2^{-d} where d is the least depth such that $s \not\approx_d t$. If we fix a constant symbol a then every closed term has a canonical representative for its $\approx_d$-equivalence class: we simply truncate the term at depth d if necessary by replacing the level-d subterm by a.

When we apply the above ideas to the closed terms that serve as names for elements of a model built by the Chase, terms that are "close" in the metric above are those that share much of their name, and intuitively behave similarly in the model. Razor uses this device to offer the user the opportunity to bound the Chase. The user can specify a *Skolem depth* d, and when the chase is about to create a new element with name t whose name-depth is greater than d, we instead re-use the element named by the depth-d term that is $\approx_d$-equivalent to t. Note that the two terms thus equated have distance less than 2^{-d} between them.

Example 13. Consider the following theory, where—in accordance with the convention in Notation 1 Skolem functions a, b and h are assigned to the existential quantifiers.

$$\vdash \exists^a \exists^b \, . \, x\, y \, . \, R(x,y)$$
$$R(x,y) \vdash \exists^h z \, . \, R(y,z)$$

Figure 2 displays three models for the previous theory, where elements are denoted by circles and tuple of relation R by edges between the elements. An unbounded run of the Chase creates the infinite model (a). Bounded runs of the Chase up to Skolem depth 1 and 2 respectively lead to models (b) and (c).

During a run of the Chase for this theory, the elements in order of creation will be named by the terms a, b, $s = h(a,b)$, $t = h(b,s)$, $u = h(s,t)$, $v = h(t,u)$, *etc.* When the search is bounded by Skolem depth 1, because $s \approx_1 t$, Razor reuses the third element to instantiate the existentially quantified variable z instead of creating a fresh element. Similarly, when the search is bounded by Skolem depth 2, because $u \approx_2 v$, the fifth element is reused

The net result is that we construct a model in which terms have been identified that may not have been equal in our chase-model. But in a precise sense, we only identify elements that are close in the sense of the metric above. This technique can also be used in an iterative-deepening approach to unbounded model-finding.

Theorem 14. *Let $\mathbb{M}$ be a model on domain $|\mathbb{M}|$ constructed by a bounded search up to depth d. Let d_1 and d_2 be the depth of $\mathrm{N}_{\mathbb{M}}(\mathbf{e_1})$ and $\mathrm{N}_{\mathbb{M}}(\mathbf{e_2})$ for two elements $\mathbf{e_1}$ and $\mathbf{e_2}$ in $\mathbb{M}$, given $d_1 < d$ and $d_2 < d$. Then $\mathbf{e_1} = \mathbf{e_2}$ is true in $\mathbb{M}$ if and only if it is justified by a pair $(\sigma, \eta) \equiv \Gamma_{\mathbb{M}}(\mathbf{e_1} = \mathbf{e_2})$ for $\sigma \in \mathcal{T}$ and $\eta : \vec{x} \to |\mathbb{M}|$, where $\vec{x}$ is the set of free variables in σ.*

We sketch a proof of Theorem 14: by Theorem 11, every fact in $\mathbb{M}$ that is constructed by an unbounded run of the Chase is justified. This includes the equational facts over the elements of $\mathbb{M}$. However, when the search is bounded by depth d, Razor reuses a canonical depth-d element, e_d, instead of creating a new element $\mathbf{e}$ when $\mathrm{N}_{\mathbb{M}}(\mathbf{e})$ has a depth greater than d. This forces an unjustified equality between e and e_d in $\mathbb{M}$.

When running Razor with a bound on the depth of Skolem terms, then the goal of “no confusion” is of course compromised. But since the unjustified equalities in those models are restricted to elements with names of the maximum depth we may say that we have “controlled confusion.” Specifically, if two terms t_1 and t_2 are unjustifiably equated by a bounded run, with bound d, then we know that the distance between t_1 and t_2 is less that 2^{-d}.

3.4.1 Filesystem Example: Bounded Model-Finding

The idea of bounding the search by the depth of element names guarantees that any run of the Chase for the Filesystem theory will terminate. Let us assume that the search is bounded by depth 2: as described above, a run of the Chase that always chooses the first disjunct of Sequent 8 creates $\mathbf{e_3}$ to be the parent of $\mathbf{e_2}$ in the filesystem $\mathbf{e_1}$. The term $\mathrm{N}_{\mathbb{M}}(\mathbf{e_3}) = \mathsf{hasParent}(\mathsf{someFileSys}, \mathsf{someObject})$ is the name of the new element in the current model. In the next iteration, this run of the Chase will create a new element $\mathbf{e_4}$ to serve as the parent of $\mathbf{e_3}$, where $\mathrm{N}_{\mathbb{M}}(\mathbf{e_4}) = \mathsf{hasParent}(\mathsf{someFileSys}, \mathrm{N}_{\mathbb{M}}(\mathbf{e_3}))$. Next, when the Chase is about to introduce a parent $\mathbf{e_5}$ for $\mathbf{e_4}$ with name $\mathrm{N}_{\mathbb{M}}(\mathbf{e_5}) = \mathsf{hasParent}(\mathsf{someFileSys}, \mathrm{N}_{\mathbb{M}}(\mathbf{e_4}))$, because $\mathrm{N}_{\mathbb{M}}(\mathbf{e_5}) \approx_2 \mathrm{N}_{\mathbb{M}}(\mathbf{e_4})$, the Chase will reuse the existing element $\mathbf{e_4}$ instead of creating $\mathbf{e_5}$. This will break the infinite cycle of creating new elements. In this example, however, $\mathit{Parent}(\mathbf{e_1}, \mathbf{e_4}) = \mathbf{e_4}$ implies $\mathit{Contents}(\mathbf{e_1}, \mathbf{e_4}, \mathbf{e_4})$, which causes the current run of the Chase to fail on Sequent 14.

4 Implementation

Scheduling We maintain a run of the Chase for a theory $\mathcal{T}$ as a pair $(\mathcal{Q}, \mathbb{M})$, called a *problem*, consisting of a model $\mathbb{M}$ and a queue $\mathcal{Q}$ of sequents. The model $\mathbb{M}$ is the collection of all facts inferred by the current run of the Chase and $\mathcal{Q}$ is used to schedule the sequents of $\mathcal{T}$ in a fair manner. In every execution of Razor, the overall state of computation is a *pool* of problems, initialized with a single problem with an empty model. In every cycle, Razor selects a problem $(\mathcal{Q}, \mathbb{M})$ from the pool, selects a sequent σ from the head of $\mathcal{Q}$, computes a model $\mathbb{M}'$ by applying a chase-step on σ for $\mathbb{M}$ and some environment η in which $\mathbb{M}$ does not satisfy σ, reschedules σ at the end of the queue yielding $\mathcal{Q}'$, and, restores $(\mathcal{Q}', \mathbb{M}')$ into the pool. A problem $(\mathcal{Q}, \mathbb{M})$ will be removed from the pool and $\mathbb{M}$ will be presented to the user once $\mathbb{M}$ is a model of $\mathcal{T}$; the problem will also be removed if the Chase fails on $\mathbb{M}$.

Performing a chase-step for a problem $(\mathcal{Q}, \mathbb{M})$ with $\sigma \equiv \varphi \vdash \bigvee_{i=1}^{n} \psi_i$ at the head of $\mathcal{Q}$ yields a set of problems $P_i = (\mathcal{Q}', \mathbb{M}_i)$ for $1 \leq i \leq n$. Each model $\mathbb{M}_i$ is the result of making a disjunct ψ_i true in $\mathbb{M}$, enabling us to follow the consequences of every choice of disjunct as a separate problem in the pool.

In practice, we use a priority queue to schedule the sequents of theory $\mathcal{T}$ in a problem: this allows the scheduler to prioritize sequents with $\bot$ in their heads over the the other sequents that may extend the current model with new facts and elements. As a consequence, Razor terminates a failing branch before wasting unnecessary chase-steps on the other sequents of the theory.

Sequents as Relational Algebra Queries Computationally, a major cost of applying a chase-step on a sequent $\sigma \equiv \varphi \vdash_{\vec{x}} \psi$ and a model $\mathbb{M}$ is due to computation of an environment η in which $\mathbb{M}$ does not satisfy σ. The cost of a naive computation of η grows significantly with the size of $\mathbb{M}$ as it requires testing σ in $\mathbb{M}$ for every map from $\vec{x}$ to $|\mathbb{M}|$. For an efficient solution to this problem, we implement the Chase from a relational algebra perspective as follows:

Via the well-known correspondence between first-order formulas and relational algebra expressions, a positive-existential formula φ can be regarded as a database *view* with a relational expression V_φ. Consequently, a geometric sequent $\varphi \vdash_{\vec{x}} \psi$ will correspond to a pair of *union-compatible* views, V_φ and V_ψ, with $\vec{x}$ as their attributes.

Let $V_\varphi(\mathbb{M})$ denote the set of tuples returned by evaluating V_φ in a model $\mathbb{M}$ as database. Then $\mathbb{M}$ satisfies the sequent $\varphi \vdash_{\vec{x}} \psi$ if and only if $V_\varphi(\mathbb{M}) \subseteq V_\psi(\mathbb{M})$ holds. Therefore, a chase-step on sequent $\varphi \vdash_{\vec{x}} \psi$ in model $\mathbb{M}$ can be seen as a procedure that inserts the tuples in $V_\varphi(\mathbb{M}) - V_\psi(\mathbb{M})$ into the view $V_\psi(\mathbb{M})$ as a *view update* problem. By adopting the relational algebra perspective, an environment η is implicitly constructed as a mapping from $\vec{x}$, to some tuple t in $V_\varphi(\mathbb{M}) - V_\psi(\mathbb{M})$.

The current implementation of Razor starts with translating the sequents of input theory to pairs of relational expressions in a preprocessing step. Accordingly, the Chase is implemented as described above. We have also been inspired by various ideas in the database literature to improve the efficiency of Razor. In particular we have developed a mechanism for *incremental view maintenance* that allows Razor to evaluate the views efficiently as the model grows.

Equational Reasoning The current version of Razor performs a naive algorithm for equational reasoning: every time a chase-step adds a new fact $\mathbf{e_1} = \mathbf{e_2}$ to a model, Razor simply collapses the two elements by rewriting all of the occurrences of $\mathbf{e_2}$ in the current model with $\mathbf{e_1}$. Due to monotonicity, the current algorithm is sound: adding more facts to the current model will not violate the equality of $\mathbf{e_1}$ and $\mathbf{e_2}$.

Developing an efficient algorithm for handling equality reasoning in our context remains as a future work.

5 Case Studies

The current version of Razor is a proof of concept, developed to explore the idea of model-finding for theories in geometric form. In particular, the purpose of the current implementation is to study the characteristics of the models constructed by a chase-based algorithm, to evaluate the termination

Spec.	**Alloy**		**Aluminum**			**Razor**		
	# models	**bound**	**# models**	**size range**	**bound**	**# models**	**size**	**Skolem depth**
Bday(2)	27	8	1	4-4	8	1	3-3	no bound
Bday(3)	11	7	1	4-4	7	1	4-4	no bound
Gene	64	6	64	6-6	6	162	6-12	2
Gpa	2	4	2	4-4	4	2	4-4	no bound
Java	1566	9	3	4-4	9	15	5-6	1
File(1)	0	10	0	-	10	0	-	no bound
Grade(1)	10837	9	3	4-5	9	2	4-5	no bound
Grade(2)	49	6	3	4-5	6	2	4-5	no bound
Grade(3)	3964	9	1	2-2	9	1	2-2	no bound

Figure 3: Models returned by Alloy, Aluminum and Razor.

results of such a model-finding solution, to study the practical implications of constructing provenance information, and to realize the idea of exploring the space of models through minimality.

We executed our model-finding tool for the sample specifications that we had previously used for evaluating Aluminum [18], after we manually converted them to the geometric form (available at `https://github.com/salmans/RazorExamples`). Although the current version has not been designed and implemented with performance considerations, the execution times of Razor in our experiments are comparable to those of Alloy and Aluminum.

Table 3 compares the number and the size of models produced by Alloy, Aluminum and Razor. The first column represents the specifications birthday (*Bday*), genealogy (*Gene*), grandpa (*Gpa*), javatypes (*Java*), filesystem (*File*) and gradebook (*Grade*). All of the sample specifications except *Grade* are taken from the Alloy distribution. In this column, the numbers in parentheses distinguish variations of a root specification. The second and the third columns respectively contains the number of models returned by Alloy and the bound size for those models. The next three columns respectively display the number of models, the range of number of elements in those models, and the total bound size for the models produced by Aluminum. (Since the number of models produced by Alloy is large in some cases, we did not compute a size range for these sets.) The same bound sizes have also been used for our experiments with Alloy. Similarly, the last three columns demonstrate the number of models, the range of number of elements in those models, and the Skolem depth used to bound the models constructed by Razor.

Weakly-Acyclic Theories When the theory is weakly acyclic (*Bday*, *Gpa*, *File*(1), *Grade*), unbounded search is possible. For these specifications, Razor returns a finite set of universal models, assuring the user that every model of the theory reduces to one of the universal models: all three model-finders, Alloy, Aluminum and Razor, return exactly two models for the specification of the well-known “I’m my own Grandpa” example (*Gpa*). However, because Alloy and Aluminum perform a bounded search (for models with up to 4 persons), it is not clear that if the given bound is sufficient to contain all scenarios where a person is his own grandpa. The unbounded search by Razor, on the other hand, reveals that in every model of the theory, a person may be his own grandpa because of two specific combinations of relations between exactly 4 persons. Put differently, the two models returned by Razor form the support of all the models of the theory.

Unsatisfiable Theories Unbounded search is reassuring when the theory is unsatisfiable: *File*(1) specifies a filesystem and tries to build a counterexample that shows moving a file in the filesystem is "unsafe." Razor guarantees that $File(1)$ is unsatisfiable, thus, moving files in the specified filesystem is in fact "safe." However, Alloy and Aluminum can show that the specification does not have any models only up to some given bound.

Homomorphic Minimality A crucial point to keep in mind is that Aluminum computes models that are minimal with respect to the property of being a submodel, as opposed to the homomorphism order. For $Grade(1)$ and $Grade(2)$, Alloy and Aluminum execute the same command but with different bounds on the size of models. In every execution, Aluminum starts with some arbitrary model, as constructed by Alloy, removes the unnecessary facts from the model until every fact in the model is inevitable for satisfying the theory. This process, however, does not guarantee a model that is homomorphically minimal as the model may contain confused elements. As for unbounded runs of Razor, however, the resulting models for these two specifications are homomorphically minimal. The extra models that are returned by Aluminum for $Grade(1)$ and $Grade(2)$, as well as the extra element in the model for $Bday(2)$ is caused by this discrepancy.

It is important to keep in mind that a run of Razor that is bounded by some Skolem depth may also produce models that contain confused elements hence not minimal. However, confusion among the elements of models that Razor builds is controlled as discussed in Section 3.3.

Bounded Model-Finding Because *Java* and *Gene* are not weakly-acyclic, Razor's search for models has to be bounded to avoid nontermination. However, because Aluminum puts a bound on the size of models but Razor's search is bounded by Skolem depth, the number of models returned by Aluminum and Razor are not comparable. Moreover, Aluminum and Razor return different sets of models for *Java* and *Gene* since confusion is arbitrary in models that Aluminum builds but controlled in those of Razor.

Conclusion

We presented Razor, a model-finder for theories in geometric form, which allows the user to explore the models of a first-order specification. The core functionality of Razor is to facilitate the two aspects of exploration, (i) exploring the space of all models of a theory through augmentation, and (ii) apprehending individual models via provenance information based on Skolemization. Razor implements a novel approach to bound the search for models, which allows for "controlled confusion" among the elements of models. We also reported on the results of our experiments with our preliminary implementation of Razor.

Acknowledgments. We are grateful to Kathi Fisler, Joshua Guttman, Shriram Krishnamurthi, Tim Nelson, and John Ramsdell for their insights and encouragement.

References

[1] Samson Abramsky. Domain theory in logical form. *Ann. Pure Appl. Logic*, 51(1-2):1–77, 1991. Second Annual IEEE Symposium on Logic in Computer Science (Ithaca, NY, 1987).

[2] Catriel Beeri and Moshe Y Vardi. A proof procedure for data dependencies. *Journal of the ACM (JACM)*, 31(4):718–741, 1984.

[3] Marc Bezem and Thierry Coquand. Automating coherent logic. In *LPAR, Logic for Programming, Artificial Intelligence, and Reasoning*, volume 3835 of *Lecture Notes in Computer Science*, pages 246–260, 2005.

[4] F. Bry and A. Yahya. Positive unit hyperresolution tableaux and their application to minimal model generation. *Journal of Automated Reasoning*, 2000.

[5] Hans de Nivelle and Jia Meng. Geometric resolution: A proof procedure based on finite model search. In Ulrich Furbach and Natarajan Shankar, editors, *IJCAR*, volume 4130 of *Lecture Notes in Computer Science*, pages 303–317. Springer, 2006.

[6] Alin Deutsch, Alan Nash, and Jeff Remmel. The chase revisited. In *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 149–158. ACM, 2008.

[7] Alin Deutsch and Val Tannen. Xml queries and constraints, containment and reformulation. *Theoretical Computer Science*, 336(1):57–87, 2005.

[8] Shaddin F. Doghmi, Joshua D. Guttman, and F. Javier Thayer. Searching for shapes in cryptographic protocols. In *Tools and Algorithms for the Construction and Analysis of Systems*, 2007.

[9] R. Fagin, P. Kolaitis, R. Miller, and L. Popa. Data exchange: Semantics and query answering. *Database Theory ICDT 2003*, pages 207–224, 2002.

[10] R. Fagin, J.D. Ullman, and M.Y. Vardi. On the semantics of updates in databases. In *Symposium on Principles of Database Systems*, 1983.

[11] Kathi Fisler, Shriram Krishnamurthi, Leo Meyerovich, and Michael Tschantz. Verification and change impact analysis of access-control policies. In *International Conference on Software Engineering*, 2005.

[12] Joshua D. Guttman. Security theorems via model theory. *EXPRESS: Expressiveness in Concurrency (EPTCS)*, 8:51, 2009. doi:10.4204/EPTCS.8.5.

[13] Daniel Jackson. *Software Abstractions*. MIT Press, 2 edition, 2012.

[14] Bernard Linsky. Logical constructions. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2012 edition, 2012.

[15] J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming*. The MIT Press, 1992.

[16] David Maier, Alberto O Mendelzon, and Yehoshua Sagiv. Testing implications of data dependencies. *ACM Transactions on Database Systems (TODS)*, 4(4):455–469, 1979.

[17] William McCune. MACE 2.0 reference manual and guide. *CoRR*, 2001. cs.LO/0106042.

[18] Tim Nelson, Salman Saghafi, Daniel J Dougherty, Kathi Fisler, and Shriram Krishnamurthi. Aluminum: principled scenario exploration through minimality. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 232–241. IEEE Press, 2013.

[19] Timothy Nelson, Christopher Barratt, Daniel J. Dougherty, Kathi Fisler, and Shriram Krishnamurthi. The Margrave Tool for Firewall Analysis. In *USENIX Large Installation System Administration Conference*, 2010.

[20] Ilkka Niemelä. A tableau calculus for minimal model reasoning. In *Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, 1996.

[21] A. Robinson. *Handbook of Automated Reasoning*, volume 2. Elsevier, 2001.

[22] Benjamin Rossman. Existential positive types and preservation under homomorphisms. In *Logic in Computer Science, 2005. LICS 2005. Proceedings. 20th Annual IEEE Symposium on*, pages 467–476. IEEE, 2005.

[23] Bertrand 1924 Russell. Logical atomism. In D. F. Pears, editor, *The Philosophy of Logical Atomism*, pages 157–181. Open Court, 1985.

[24] Steven Vickers. Geometric logic as a specification language. In Chris Hankin, Ian Mackie, Rajagopal Nagara-janChris Hankin, Ian Mackie, and Rajagopal Nagarajan, editors, *Proceedings for the Second Imperial College Department of Computing Workshop on Theory and Formal Methods*, pages 321–340, 1995.

[25] J. Zhang and H. Zhang. SEM: a system for enumerating models. In *International Joint Conference On Artificial Intelligence*, 1995.