On ∃ ∀ ∃! Solving: A Case Study on Automated Synthesis of Magic Card Tricks

Susmit Jha Vasumathi Raman
United Technology Research Center, Berkeley
jhask, ramanv@utrc.utc.com

Sanjit A. Seshia
EECS, UC Berkeley
sseshia@eecs.berkeley.edu

***Abstract*—Many techniques have been proposed for automated synthesis of systems from components. In formal synthesis, we need to find a composition of components from a finite library such that the composition satisfies a given logical specification. In this paper, we consider the problem of synthesis of magic card tricks from component actions, where some of the actions depend on non-deterministic choices. This problem can be naturally represented as a quantified logical formula of the form: *Exists* a composition, *Forall* nondeterministic choices, *Uniquely-Exists* intermediate and final outputs satisfying a logical specification, that is, an (∃∀∃!) satisfiability problem. We present an approach to solve this problem using a novel approach that exploits the unique-existence of intermediate and final outputs for any given composition and choice values. We illustrate how popular magic card tricks can be designed using this approach. These tricks evolved through human ingenuity over decades, but we demonstrate that formal synthesis can generate a number of novel variants of these tricks within minutes. In contrast, a direct encoding to quantified SMT problem fails to find a solution in hours.**

I. Introduction

Composition has played a central role in enabling configurable and scalable design of efficient systems across different domains. Automated formal synthesis of systems as composition of elementary components using satisfiability solving techniques has proved to be useful in creating non-intuitive artefacts such as bit-vector programs using bit-twiddling operations [1] and deobfuscated code [2]. Synthesis from components can also be carried out by selecting a syntactically expressible family of compositions such as a sketch with a fixed skeleton but free choice of components [3], [4]. But these approaches are restricted to deterministic functional components. In this paper, we consider the problem of synthesizing composition of components where they represent primitive actions such as shuffling or cutting card deck or turning over of cards. Some of these component actions are nondeterministic and their output depends on uncontrollable external choices. This formulation can be used for synthesis of strategies against unknown non-deterministic choices made by the adversary or synthesis of circuits with stochastic (abstracted as non-deterministic) components, in addition to the application to synthesizing magic card tricks considered in this paper. Magic card tricks are often developed over decades and use significant human ingenuity. These tricks involve interesting application of number theory and discrete mathematics, and hence, their explanation has received significant attention from mathematicians [5], [6]. In particular, they have been found to be useful in teaching computer science [7]. To the best of our knowledge, this is the first application of formal methods to synthesize new magic card tricks. Beyond this unconventional application to humanities and an attempt to automate a long-standing bastion of human creativity, we believe this application will also add interesting benchmarks to fuel further work on automated synthesis of systems with non-deterministic components.

In previous work [1], we have shown that the synthesis of composition using components can be expressed in first-order logic by eliminating the existential quantification over the predicate corresponding to composition using integer parameters to encode the set of possible compositions for a finite library of components. We make the following novel contributions in this paper:

- The synthesis problem using non-deterministic component actions is directly representable as ∃∀∃! problem and it is shown to have a corollary of the ∃∀ form. This dual representation enables the use of counterexample guided inductive synthesis.
- We model different component actions in magic card tricks logically in Z3 SMT solver [8] and discover novel magic card tricks as composition of these actions. While human discovery of these variants takes decades, we discover new magic tricks within minutes.

In the rest of the paper, we first present a brief background on magic card tricks in Section II. We present the formulation of synthesis of magic tricks as a composition synthesis problem and its reduction to the satisfiability problem in Section III. We present the results including new magic tricks discovered by our approach in Section IV. We conclude with discussion of future work in Section V.

II. Background on Magic Card Tricks

Magic card tricks consist of a sequence of actions in which the performer manipulates the cards through acts such as shuffling or turning over cards. Usually, these action sequences also involve audiences making a number of choices independently based on their discretion, and sometimes in secret without the performer knowing the exact choice of the audience. These choices are intended to make the audience believe that they are in control of the manipulation of cards. Consequently, the performer's ability to achieve some deterministic goal such as

finding a particular card or ensuring that the cards end up in a particular specific pattern surprises the audience and makes the entire process appear *magical*.

The secret sauce behind these magic card tricks often lies in some mathematical invariant of the sequence of actions which allow the performer to ensure a deterministic output irrespective of the exact choice made by the audience. Such sequence of actions are naturally hard to generate, and consequently, each sequence is passed on as training. We use an example of a very simple magic card trick to give insight into the underlying mathematics that ensures the outcome of the trick is independent of audience choices. This example magic trick was invented by magician Bob Hummer and popularized as Face Up/Face Down Mysteries (1941)[9]. It has the following sequence of actions.

1) Take any ten cards and have them all face-down and hold them as you are going to deal the cards.
2) Go through the following procedure which mixes the cards face-up and face-down based on audience choice: Spread the top two cards off and turn them over together, placing them back on top. Let an audience give the deck of cards a straight cut [1]. The cut can be at any position (odd or even) of the deck. Repeat this "turn-two and audience-directed cut" at random as often as you like. The cards will be in an *apparent* unpredictable mess.
3) To find the order in the mess, proceed as follows: Go through the card deck, reversing every second card (the cards in positions 2, 4, 6, 8, and 10). You will find exactly five cards face-up, no matter how many times the "turn two and cut at random" procedure was repeated.

The ten cards with faces-up or down can end up in $2^{10} \times 10!$ possible arrangements which is more than 3.6 billion. The *magical* surprise comes out of the fact that we always end up in a state satisfying a deterministic property irrespective of the choices made by the audience. The explanation for this lies in the following observation. The actions in step 2 (also called Hummer Shuffle) which include choice from audience has the following invariant - the number of cards facing up/down at even places in the stack is respectively equal to the number of cards facing up/down at odd places after each Hummer Shuffle. Let m be the number of face-up cards in even places, and by the above invariant, also in odd places. The last step of turning over even cards will leave us with $5 - m$ face-up cards in even places. Combined with m face-up cards in odd places, the total number of face-up cards will always be five irrespective of the audience choices.

A more popular derivative of this trick called Royal Hummer invented by Steve Freeman decades later is used to produce royal flush from a deck of cards that has been apparently shuffled randomly by the audience. The Hummer Shuffle consisting of just two actions: turn two and audience-directed cut, was invented more than seventy years ago but it still remains the core of many magic card tricks. Our goal is to automatically synthesize even longer (and hence, even more surprising and non-intuitive) sequences which are guaranteed to satisfy some relevant deterministic property on the final state irrespective of the choices made by the audience.

[1]Audience chooses a random position in the deck and all the cards below this position are put contiguously on the top of the deck. See [9], [10] for details of card trick terms.

III. Composition Based Synthesis of Card Tricks

Automated formal synthesis of systems (particularly programs) from high-level specification has received significant attention over the past decade. For brevity, we refer to a recent paper [3] and references therein for detailed discussion on state of the art in synthesis. We focus on the problem of formal modelling and synthesis of magic card tricks in this section. Given a deck of cards, we can characterize the state of cards as a state $s \in S$ where S is all the possible configurations the cards can be put in. The states would capture relevant details for a trick such as whether a card is facing up or down, the position of each card and color or kind of cards in each position. For example, the Hummmer trick described in Section II, the state space is given by whether the cards are facing-up or down in each of the 10 positions. We denote the library of primitive actions, such as turning over a card or cutting the card deck, by $L = \{A_1, A_2, \ldots, A_n\}$ where each action is either deterministic or non-deterministic. For notational uniformity, we associate both kinds of actions A_i to a state transformation function $T_i : S \times C \to S$ where C is the set of choices. Execution of actions A_i takes the cards from state s to $s' = T_i(s, c)$ where $c \in C$ is the choice parameter. Deterministic actions ignore the second argument, that is, the choice parameter. In order to generate action sequences of varying length, we include k copies of noop actions $A_{n+1} \ldots A_{n+k}$ in the library L such that the transformation function for each noop action is identity, that is, $T_{n+1}(s, c) = \ldots T_{n+k}(s, c) = s$. A schematic of magic trick as a sequence of actions is shown in Figure 1.

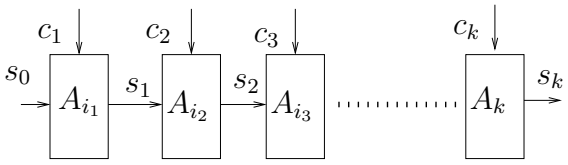

Fig. 1. Magic Scheme as Sequence of Actions

From the scheme presented above and using the first-order representation of connection predicate [1], [2], the problem of synthesizing magic trick can be formulated as follows, assuming a unique initial state s_0 (generalization to multiple initial states can be easily done). We assume that k is the upper bound on the length of the magic trick. $k \leq n$ and hence, a trick may not use all actions in the library. Further, we include k copies of noop action with identity state transformation function, and thus, the sequence may use these functions which can be eliminated as post-processing to generate shorter magic sequences.

Let the vector $i_1 i_2 \ldots i_k$ corresponding to the sequence of actions, where $i_j \in [0, n]$ selects one of the actions, be denoted

by conn, $c_1, c_2, \ldots c_k$ with $c_j \in C$ be denoted by choices, and $s_1, \ldots s_k$ with $s_j \in S$ be denoted by states. Further, the function T is a parametrized state transformer which uses i_m as its first argument to select the corresponding action A_{i_m} and thus, the function T_{i_m}. We can now rewrite the above constraints as

$\mathcal{F}_{des}$: $\exists$conn $\forall$choices $\exists!$states

$$\phi_{des}(\texttt{conn}, \texttt{choices}, s_0, \texttt{states}) \wedge \phi(\texttt{states})$$

where $\phi_{des}(\texttt{conn}, \texttt{choices}, s_0, \texttt{states}) = \bigwedge_{m=1}^{k}(s_m = T(i_m, s_{m-1}, c_m))$ and $\phi(\texttt{states}) = \phi_{spec}(s_k)$ is the deterministic requirement on the final state. Since the possible choices constitute a finite (albeit a very large) set, say $\{\texttt{choices}_1, \texttt{choices}_2, \ldots, \texttt{choices}_\texttt{L}\}$, and there exists exactly one state vector $\texttt{states}_l$ for each choice vector $\texttt{choices}_l$, we can rewrite $\mathcal{F}_{des}$ as follows:

$\mathcal{F}_{sat}$: $\exists$conn $\exists\texttt{states}_1 \exists\texttt{states}_2 \ldots \exists\texttt{states}_\texttt{L}$

$$\bigwedge_{l=1}^{L} \phi_{des}(\texttt{conn}, \texttt{choices}_l, s_0, \texttt{states}_l) \wedge \phi(\texttt{states})$$

The challenge with solving $\mathcal{F}_{sat}$ is that the number of choices is huge. But the above formulation can be used to find a sequence of actions conn that is consistent with a subset of example choice vectors. Let us consider another constraint:

$\mathcal{F}_{ver}$: $\exists$conn $\forall$choices $\forall$states

$$\phi_{des}(\texttt{conn}, \texttt{choices}, s_0, \texttt{states}) \Rightarrow \phi(\texttt{states})$$

We observe that $\mathcal{F}_{ver}$ is true if $\mathcal{F}_{des}$ is true, that is, $\mathcal{F}_{des} \Rightarrow \mathcal{F}_{ver}$ due to the unique existence of the states for any given choice vector in $\mathcal{F}_{des}$. For any given sequence of component actions conn, a counterexample obtained by solving the satisfiability problem for $\exists\texttt{choices}\exists\texttt{states}\ \neg(\phi_{des}(\texttt{conn}, \texttt{choices}, s_0, \texttt{states}) \Rightarrow \phi(\texttt{states}))$ can be, thus, used to solve the original synthesis problem in $\mathcal{F}_{des}$ using component-based program synthesis [1], [2]. The generalization from examples is done by solving the satisfiability problem for $\mathcal{F}_{sat}$ including only the choices found from solving $\mathcal{F}_{ver}$.

IV. Results

In this section, we present the results on automated synthesis of magic card tricks. We consider three different known magic card tricks [9], [10] and we synthesize a number of novel variants for each of these tricks. These novel variants are new sequence of actions which ensure that the final state meets the given deterministic requirement irrespective of the choices made by the audience for non-deterministic actions. Our experiments used Z3 [8] SMT solver for checking satisfiability and all experiments were run on a 2.9 GHz Intel Core i7 with 16 GB RAM. Finding the action sequence for the example presented in Section II took 3 minutes 18 seconds.

A. Baby Hummer

In this magic trick, the audience selects a card unknown to the magician, and three other cards creating a card deck of 4 cards with the selected card at the bottom. All the cards are put in the deck facing down. The magician instructs the audience to take a sequence of actions - some of which depend on the choice of the audience, such that the card deck appears to be randomly shuffled. At the end, the magician states that all the cards in the deck except one must face in the same direction - either up or down. The deck must have only one odd-facing card, and this card is the one selected by the audience. We build a glossary of actions used in this trick before describing the actions in the original card trick and the synthesized novel variants. The finite library used in the automated synthesis process consists of four copies of each action listed below, and noop actions as described in Section III.

Action	Description
turntop	Turn over the face of the top card.
turntop2	Take off the top two cards (keeping them together), turn them over together and place them back on top.
toptobottom	Place the top card of the deck to the bottom.
top2tobottom	Place the top two cards of the deck to the bottom keeping them together and in-order
cut	Audience-choice directed straight-cut of the card deck.

The sequence of actions in the original magic card trick are as follows:

- toptobottom, turntop, cut, turntop2, cut, turntop2, cut, turntop2, turntop, top2tobottom, top2tobottom, turntop

The reader can refer to Fig 4-13 in Chapter 1 of [9] for picture illustrations of this magic trick.

We used the synthesis approach presented in Section III to generate new action sequences of at most length of 15 steps. Our initial experiments yielded non-interesting sequences where the cut action was never used. We added constraints to ensure that cut action is always included in the sequence. This led to sequences where all the cut were either at the start of the sequence or at the end of the sequence. These are also not interesting since the cuts do not turn the cards and hence, they don't achieve any shuffling of face-up and face-down cards unless mixed with turning-over actions.

After adding constraints to mix the card-turning actions with cut, we generated a number of new sequences which also ensure that the audience-selected card is the odd-facing card at the end, irrespective of the choices made by audience. We report six of these (with the noops removed) below:

- toptobottom, turntop, cut, cut, turntop, toptobottom, turntop2, toptobottom, turntop, toptobottom
- turntop2, turntop, toptobottom, toptobottom, cut, turntop, toptobottom, toptobottom, turntop, cut, turntop2, turntop2
- turntop2, toptobottom, toptobottom, turntop, cut, toptobottom, cut, turntop2, turntop2, cut, turntop, turntop
- toptobottom, turntop, cut, toptobottom, toptobottom, turntop2, cut, cut, turntop, toptobottom, turntop, turntop2
- toptobottom, turntop, cut, toptobottom, toptobottom, cut, turntop2, cut, turntop, toptobottom, turntop, turntop2
- toptobottom, toptobottom, toptobottom, turntop, cut, top2tobottom, cut, turntop2, turntop2, cut, turntop, turntop

The runtime for synthesis was 12m 23sec, 12m 04sec, 10m 30sec, 5m 43 sec, 11m 11 sec and 11m 39 sec. respectively. A direct $\exists\forall\exists$ encoding of the problem timed out after 2 hours and could not find any action sequence.

B. Elmsley Shuffles

Shuffles are very common component actions of many magic card tricks. We consider two perfect shuffle actions: in-shuffle and out-shuffle. Both in-shuffle and out-shuffle begin by splitting the deck into two equal piles by splitting it in the middle, followed by shuffling perfectly so that the new deck has cards alternately from both piles. The original top card is the second card in in-shuffle and it is the top card in out-shuffle. In this magic trick, the performer asks the audience to randomly shuffle the cards and select one card which is put on top of the stack but not told to the performer. The performer then does a number of in-shuffles and out-shuffles which make the stack appear to be randomly shuffled but the performer is able to take out the card from a particular position in the stack, and this card is the one selected by the audience. Since these shuffles do not depend on the initial state of the card deck, simple enumeration could also work in this case. But more interesting shuffles will not be possible through enumeration. Let `inS`, `outS` denote the two actions of in-shuffle and out-shuffle respectively. Starting with 8 cards, two action sequences that would put the top card at position 6 are as follows: `inS`, `inS`, `outS`; and `inS`, `outS`, `outS`, `outS`, `inS`, `outS`. Similarly, two action sequences which would put the top card at position 5 are as follows: `inS`, `outS`, `inS`; and `inS`, `outS`, `outS`, `outS`, `outS`, `inS`. The runtimes for synthesis were 6m 42sec, 8m 52sec, 5m 18sec and 6m 08sec.

C. "Mind Reading" of Cards

In this magic trick, the performer starts with a prearranged card deck with 8 cards which are passed to the audience. Let this sequence be AH, 5D, 6H, 2S, 5S, KC, 7H, 8S where H,D,C,S stands for hearts, diamonds, clubs and spades. The color sequence for this stack is RRRBBBRB. Any three audience members are asked to make a sequence of random cuts on this stack after which each of them select (in order) the top card as their chosen card in secret from the performer. The performer is required to find these cards. He does so by asking the audience with red cards to stand up. Depending on the audience who stand up, the performer can tell the exact card selected by the audience. This trick works because any cyclic subsequence with length 3 of RRRBBBRB is unique. So, when the audience members stand up, the performer can find the subsequence (RRR, RRB, RBR, RBB, BRR, BRB, BBR, BBB). Once this subsequence is known, the performer uses his knowledge of the prearranged stack to tell the exact cards picked by the audience. The automated synthesis requires coming up with this initial prearranged card deck. The subsequent actions performed by the audience are fixed in this trick. Clearly, not all prearranged card decks will work. The patterns needed for this trick are de Bruijn sequences also used in digital recording pens and robot localization [11]. For the variant of this trick with 5 audience members and a card deck with 32 cards (there are 32 possible sub-sequences of length 5 which can be possibly decoded using the color pattern for the 5 audiences), we synthesized the following possible color patterns. The runtime for this search was 8 m 34s and 9m 18s respectively. Note that the exact card sequence is not important and any cards consistent with these color patterns can be used as long as the performer remembers the original sequence to be able to tell the exact cards at the end irrespective of the choice of cuts by the audience.

- RRRRRBRBRRBRRRBBBBRBBBRRBBRBRBB
- RRRRRBRRBRBBRRBBBBRRRBBRBBBRBRB

V. Conclusion

In this paper, we consider the problem of synthesis of composition of non-deterministic components and show how its solution can be used to generate new magic tricks. We believe computer-aided design of magic tricks can be used as interesting examples in computer science courses. Classroom exercises on automatically synthesizing new magic card tricks can be used to introduce combinatorial search, quantified satisfiability solving and formal methods. We also plan to improve the quality of our results by adding constraints to rule out subsequences being identity operations without user-input. Efficient encoding of such constraints will also be useful in excluding dead code in program synthesis. We also plan to investigate more complex magic tricks including probabilistic tricks that rely on chance and synchronicity such as Kruskal's count trick [12]. Further, the logical modelling of magic tricks is similar to those used for computer programs from component function and for plans from component actions in artificial intelligence. In future work, we plan to investigate the application of this approach to planning in presence of adversarial choice modelled as non-deterministic component actions.

Acknowledgement: We thank Ashish Tiwari, Sonali Sinha and Claudio Pinello for very helpful discussions.

References

[1] S. Gulwani, S. Jha, A. Tiwari, and R. Venkatesan, "Synthesis of loop-free programs," in *PLDI*, 2011, pp. 62–73.

[2] S. Jha, S. Gulwani, S. A. Seshia, and A. Tiwari, "Oracle-guided component-based program synthesis," ser. ICSE '10, 2010, pp. 215–224.

[3] R. Alur, R. Bodik, G. Juniwal, M. M. K. Martin, M. Raghothaman, S. A. Seshia, R. Singh, A. Solar-Lezama, E. Torlak, and A. Udupa, "Syntax-guided synthesis," in *FMCAD*, October 2013.

[4] A. Solar-Lezama, L. Tancau, R. Bodk, S. A. Seshia, and V. A. Saraswat, "Combinatorial sketching for finite programs." in *ASPLOS*, 2006, pp. 404–415.

[5] M. Gardner, *Mathematics, magic and mystery*. Courier Corp., 2014.

[6] C. Mulcahy, "Mathematical card tricks," *Whats New in Mathematics*, 2000.

[7] J. F. Ferreira and A. Mendes, "The magic of algorithm design and analysis: Teaching algorithmic skills using magic card tricks," ser. ITiCSE '14, 2014, pp. 75–80.

[8] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, pp. 337–340.

[9] P. Diaconis and R. Graham, *Magical Mathematics: The Mathematical Ideas That Animate Great Magic Tricks*. Princeton University Press, 2011.

[10] J. Hugard and J. J. Crimmins, *Encyclopedia of card tricks*. Courier Corp.

[11] J. Pagès, J. Salvi, C. Collewet, and J. Forest, "Optimised de bruijn patterns for one-shot shape acquisition," *Image and Vision Computing*, vol. 23, no. 8, pp. 707–720, 2005.

[12] S. Humble, "Magic math cards," *The Mathematics Enthusiast*, vol. 5, no. 2, pp. 327–336, 2008.