White Dots *do* Matter: Rewriting Reversible Logic Circuits

Mathias Soeken[1] and Michael Kirkedal Thomsen[2,⋆]

[1] Group of Computer Architecture, University of Bremen
`msoeken@informatik.uni-bremen.de`

[2] DIKU, Department of Computer Science, University of Copenhagen
`shapper@diku.dk`

Abstract. The increased effort in recent years towards methods for computer aided design of reversible logic circuits has also lead to research in algorithms for optimising the resulting circuits; both with higher-level data structures and directly on the reversible circuits. To obtain structural patterns that can be replaced by a cheaper realisation, many direct algorithms apply so-called *moving rules*; a simple form of rewrite rules that can only swap gate order.

In this paper we first describe the few basic rules that are needed to perform rewriting directly on reversible logic circuits made from general Toffoli circuits. We also show how to use these rules to derive more complex formulas. The major difference compared to existing approaches is the use of *negative controls* (white dots), which significantly increases the algebraic strength. We show how existing optimisation approaches can be adapted as problems based on our rewrite rules.

Finally, we outline a path to generalising the rewrite rules by showing their forms for *reversible control-gates*. This can be used to expand our method to other gates such as the controlled-swap gate or quantum gates.

Keywords: Reversible logic, term rewriting, circuit optimisation, circuit equivalence.

1 Introduction

When Landauer presented his seminal paper [9] he exemplified some of his ideas with a simple reversible logic gate; a gate that is equivalent to the controlled-controlled-not gate or Toffoli gate. Fredkin and Toffoli [7,17] later formalized the computational model for reversible logic and reversible logic circuits have since been associated with low-power computing circuits.

An important aspect of reversible logic design is to be able to find an implementation of a desired functionality. This problem has been researched at many different levels: from hand-made (arithmetic) circuits (*e.g.* [3,5,16,19]), over different variants of synthesis algorithms (*e.g.* [11–13]), to specification languages that can ease the implementation phase (*e.g.* [14,15,20]). But another, just as

⋆ Alphabetically sorted author list.

important, aspect is to have a good or (even better) optimal implementation of the reversible circuit. Of course, some synthesis methods seek optimality, but for larger functions this is hard to achieve. This has lead to the development of algorithms for optimising reversible circuits, where some of these work directly on reversible logic circuits: these include *moving rules* [10] (there referred to as *passing rule*), for locally swapping gates without changing their functionality, and *template matching* [11], that over several gates can recognise sub-circuits which are functionally equivalent to smaller circuits.

In this work the goal is *not* to design a new and better optimisation algorithm. Instead, we desire a better understanding of the reversible circuit constructions. We do this by exploring which basic rules are necessary to perform transformations directly on reversible logic circuits. At first, we limit ourselves to general (mixed-polarity multiple-controlled) Toffoli gates. In this small, but widely used, subset of reversible gates, it is still possible to show many of the interesting features that this approach gives. In particular the use of *negative controls* (white dots) significantly increase the algebraic strength; this is shown by the power of the very simple rules. Furthermore, we also show how to use these rules to derive more complex formulas that can then be used to derive more rules.

A particular use of these rules (or algebraic laws) is in the implementation of a term rewriting system, *cf.* [4]. One use of rewriting is for optimisation, which gives a connection to template matching and moving rules. We will show this connection by deriving some template transformations and simple moving rules. Another use of rewriting is to do equivalence checking. Here an example will show how a circuit cascaded with the inverse of another can be rewritten into the identity circuit. We will only shortly discuss (but not present) a term rewriting system based on our rules. Implementing term rewriting systems is not a simple task and, among other, avoiding divergence poses a problem. The rewriting strategy based on the general decomposition of reversible circuits into V-shaped target lines [18].

Finally, it is possible to generalise the basic rules to cover other reversible logic gates than the general Toffoli gates, *e.g.* a controlled-swap (Fredkin) gate or quantum gates. We will exemplify this by defining a general control-gate and show some of the rules for these.

Some, but not all of our rules, have been used in previously presented papers, often without explicitly considering them rules. The most related work is [2], in which rules are used in order to optimise reversible circuits. However, they have either used very simple rules or rules that require the simulation of the functionality which is afterwards optimised based on Karnaugh maps and the approach is therefore not efficient for large functions. In contrast, the rules presented in this paper can all be applied structurally, hence, the size of the circuit does not matter. Also in [8] a rule-based approach is presented, but their rules only apply for a small subset of the reversible circuits (they call them *quantum Boolean circuits*), where only one single line is semantically updated: reversible many-to-one functions. In comparison our approach applies to all reversible circuits.

2 Reversible Logic and Circuits

In this paper we use the formalism of Toffoli and Fredkin [7, 17] to describe *reversible logic circuits* with diagram-notation based on Feynman [6]. In general, a *reversible gate* is defined as a bijective function from n to n Boolean values. There exist many such gates and in this work we restrict ourselves to mixed-polarity multiple-controlled Toffoli gates.

Definition 1. *Given a set of variables $X = \{x_1, \dots, x_n\}$, a* mixed-polarity multiple-controlled Toffoli gate *(referred to as* Toffoli gate *in the following) is defined as a tuple (C, x_t) of* control lines C *and* target line x_t *such that*

$$C \subset \{x, \bar{x} \mid x \in X\} \quad \text{and} \quad \{x, \bar{x}\} \not\subseteq C \text{ for all } x \in X$$

and

$$\{x_t, \bar{x}_t\} \cap C = \emptyset \enspace .$$

Control lines x and $\bar{x}$ are referred to as positive *and* negative*, respectively. From the control lines the* control function *of the gate $f : \mathbb{B}^{n-1} \to \mathbb{B}$ is defined as*

$$f : (x_1, \dots, x_{t-1}, x_{t+1}, \dots, x_n) \mapsto \bigwedge_{c \in C} c \enspace .$$

The gate represents the Boolean function $g : \mathbb{B}^n \to \mathbb{B}^n$ with

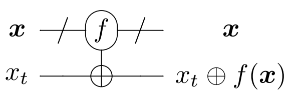
$$g : (x_1, \dots, x_n) \mapsto (x_1, \dots, x_{t-1}, x_t \oplus f(\boldsymbol{x}), x_{t+1}, \dots, x_n)$$

with $\boldsymbol{x} = x_1, \dots, x_{t-1}, x_{t+1}, \dots, x_n$.

In other words, we restrict to reversible gates with a single updated target line, x_t, and multiple control lines that are inputs to a control function f. Here f is defined as the conjunction of its inputs, where each input can be identity, negated, or not used at all. The target line is updated with the result of the exclusive-or product of x_t and the result of f. Although f cannot be any Boolean function, it is possible to define any Boolean function as an *exclusive-sum of products* (ESOP) of control functions, which will result in a *reversible circuit* with several cascaded gates that have the target line in common.

A *reversible circuit* is a cascade of reversible gates and the function defined by the reversible circuit is defined as the composition of all functions defined by the gates. It has been shown that every reversible function can be represented by a reversible circuit consisting of Toffoli gates [17].

Notation. Our notation is based on the widely-used diagrams that were introduced by Feynman [6]. The notation is illustrated in Fig. 1 by means of an example for the reversible circuit $(\{x_1, \bar{x}_2\}, x_3), (\{x_3\}, x_1), (\{\bar{x}_1\}, x_2), (\emptyset, x_3)$. The target line is marked with $\oplus$, positive and negative control lines with $\bullet$ and $\circ$, respectively. To accommodate a more general use, we will denote Toffoli gates as:



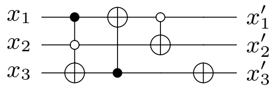


Fig. 1. Reversible circuit

Lines marked with '/' indicate that this line may consist of several bits. Note here that the control function f is not (necessarily) a reversible function (following from Def. 1), but the entire gate is still reversible as it is constructed as *reversible update* [21].

3 Rewriting Toffoli Circuits

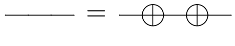
In this section we will show how to rewrite Toffoli circuits and define the basic rewrite rules that are needed. We will show how to use these rules to derive some well-known formulas, which can then be used to rewrite the Toffoli circuits in fewer steps. When performing rewrites one has infinitely many possible ways to apply the rules. It is, therefore, often helpful to have guiding strategies and we will also outline some of them here.

3.1 Rewrite Rules

We now introduce the rewrite rules. Most rules are very simple, but this is also the intention. They should contain exactly enough for us to later derive more advanced rules and this is what we will do in Sect. 3.2.

First, however, note that gate composition is associative. By this we mean that in a cascade of gates the order in which we look at the gates does not matter; *e.g.* in Fig. 1 we are free to either look at the two first gates and perform some rewriting on these, or start with the middle or last two gates. Just as we assume the existence of identity gates between all other gates.

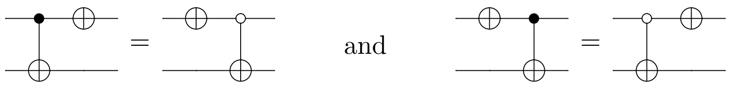
Gate Modifying Rules. The first rule is for introducing and eliminating Not-gates and states that we can always rewrite the empty line (the identity function) to two Not-gates. This is true because the Not-gate is self-inverse.



(R1)

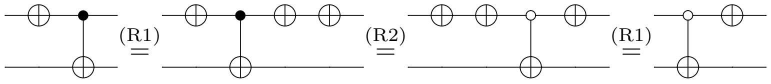
Although simple, the rule is very useful as we will see in a moment.

But before this, we will introduce the first rules with negative controls. The rule (there are two rules to be exact) simply states that we can "move" a Not-gate over a control if we negate this control.



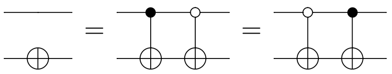
(R2)

Notice that only one of the rules is elementary and the other one can be derived from it:



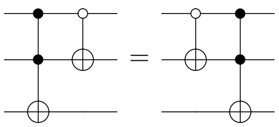
With these simple rule we, for the first time, see the power of having negative controls. Without negative controls a similar rule would not exist, which will result in a less powerful rewriting system.

Control Aware Rules. We can always extend a gate by copying it and adding once a positive and once a negative control line to it. Conversely, if two adjacent gates are equal except for one control line in which the polarity differs, the gates can be merged and the control on that line can be removed.



(R3)

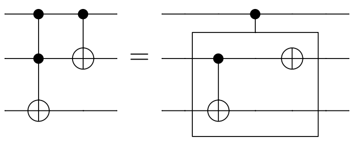
Furthermore, two arbitrary adjacent gates, which can also have different target lines, can be interchanged whenever they have a common control line with different polarities. Then, at most one of the gates can be applied.



(R4)

Notice, that the diagram notation only depicts one special case of the rule. To capture this rule in the diagrams we would need a more general notation, which will only be introduces in the generalisation (*cf.* Sect. 5).

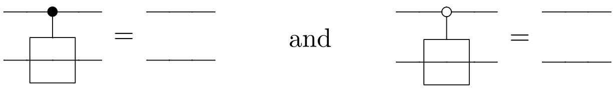
Grouping Rules. Whenever two gates share the same control line with the same polarity, these two gates can be grouped together where the group is controlled by that control line, *e.g.*



(R5)

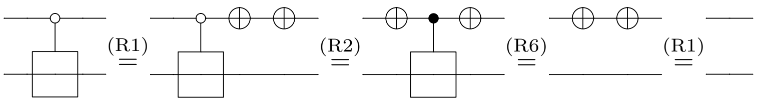
Also this rule is more general than to what is depicted in the diagram notation.

The next rule is for introducing and eliminating groups of wires. A group of wires is either zero or more gates that are controlled by the same wire; it is analogous to parenthesis in Boolean logic and can be used to work on smaller parts of the circuits. These two rules state that either a positive or negative control on a group that only contains the identity gate is equal to the identity.

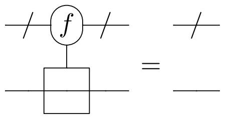


(R6)

Similar to Rule R2, one rule can be derived from the other one:

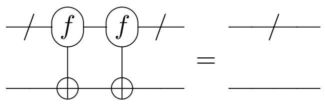


These two rules can be generalised to arbitrary control functions, *i.e.*



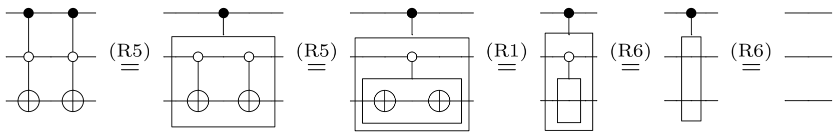
The proof is by structural induction, but we will not show it here.

Deriving More General Rules. Based on the rules described above, we are now introducing more general rules that apply to arbitrary Toffoli gates. Based on Rules R1, R5, and R6 we can derive what is famously known as *deletion rule*, *i.e.* two adjacent equal Toffoli gates can be removed:

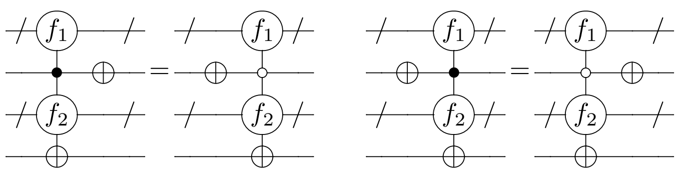


(D1)

As f is defined as conjunction of is inputs (Def. 1), the proof is by structural induction with Rule R1 being the base case and Rule R5 applied in the inductive step. The proof illustrated by the following example:

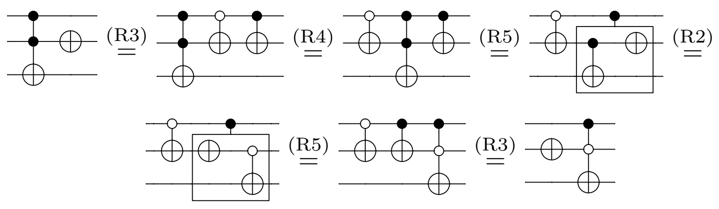


With Not-gates the control line of an arbitrary Toffoli line can be negated:

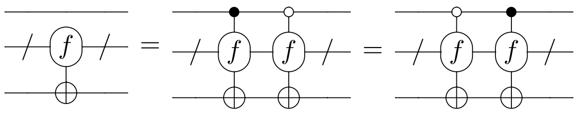


(D2)

This can also be proved using structural induction and the proof is sketched by means of the following example.



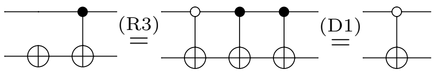
Similarly we can also generalize Rule R3 for arbitrary Toffoli gates:



(D3)

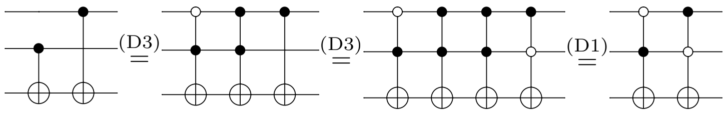
3.2 Derived Formulas

We now have seen how to use the basic rules to derive more general forms of these rules. Now, we will use the rules to derive new formulas; some of which are well-known from Boolean logic. We will use these formulas later in the paper, but at the same time it also gives more examples of using the rewrite rules. In an EXOR expression, the polarity of the operands can be swapped, *i.e.* $\bar{x} \oplus y = x \oplus \bar{y}$. This formula can also be expressed using the above introduced rewrite rules.



(D4)

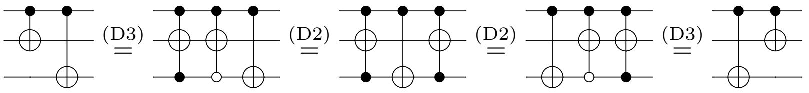
Also the following rewrite rule turns out to be quite helpful.



(D5)

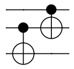
As can be seen, new rules can be composed solely from other derived rules which shows the strength of the underlying formalism.

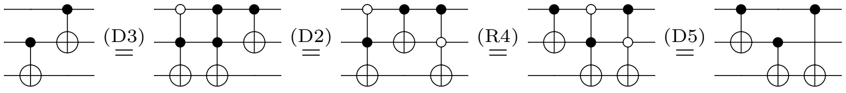
Moving Rules. Also the classical moving rule can be derived from other rewrite rules:



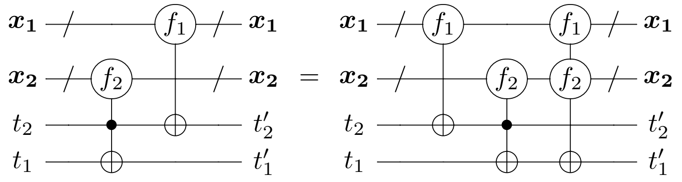
(D6)

The general moving rule can be applied to any two adjacent gates (C_i, t_i) and (C_{i+1}, t_{i+1}) if and only if $\{t_i, \bar{t_i}\} \cap C_{i+1} = \emptyset$ and $\{t_{i+1}, \bar{t}_{i+1}\} \cap C_i = \emptyset$, *i.e.* controls cannot be on wires where the other gate has a target.

But the rewrite rules are more powerful than the moving rule and allow us to interchange gates such as  for which moving rules are not sufficient:



We can generalize this rule to

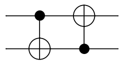


(D7)

This can also be proven using Boolean logic. On the left-hand side we have $t'_1 = t_1 \oplus (t_2 \wedge f_2(\boldsymbol{x_2}))$ and $t'_2 = t_2 \oplus f_1(\boldsymbol{x_1})$. Clearly, on the right-hand side we also have $t'_2 = t_2 \oplus f_1(\boldsymbol{x_1})$. For t'_1 we have

$$\begin{aligned} t'_1 &= t_1 \oplus ((t_2 \oplus f_1(\boldsymbol{x_1})) \wedge f_2(\boldsymbol{x_2})) \oplus (f_1(\boldsymbol{x_1}) \wedge f_2(\boldsymbol{x_2})) \\ &= t_1 \oplus (t_2 \wedge f_2(\boldsymbol{x_2})) \oplus (f_1(\boldsymbol{x_1}) \wedge f_2(\boldsymbol{x_2})) \oplus (f_1(\boldsymbol{x_1}) \wedge f_2(\boldsymbol{x_2})) \\ &= t_1 \oplus (t_2 \wedge f_2(\boldsymbol{x_2})) \end{aligned}$$

Limitations. The proposed rewrite rules cover almost all combinations in which two adjacent gates can occur. However, there is one combination that cannot be rewritten on its own. Interchanged gates with one control line such as



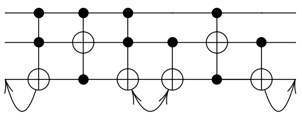
do not match any of the rules introduced above. However, they can often be rewritten if they occur as a sub-circuit in a larger circuit, which is *e.g.* shown later in Sect. 4 when rewriting the template with the id 5.1.

3.3 Rewriting Strategies

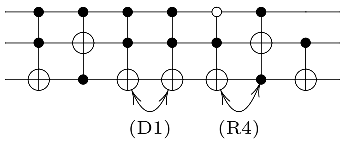
In order to implement an algorithm that makes use of the rewrite rules, a good rewriting strategy is inevitable as otherwise convergence is not necessarily guaranteed. As one rewriting strategy we suggest to bring the circuit into a V-shape. Gates in V-shaped circuits are arranged in a way that the target line for each gate moves down from the top to the bottom and afterwards moves up again. Adjacent gates can have their target on the same line. That each reversible function can be represented as V-shaped circuit has *e.g.* been shown in [12]. In their paper, Algorithm B describes a synthesis algorithm that can be applied to all reversible functions and naturally results in V-shaped circuits.

4 Examples

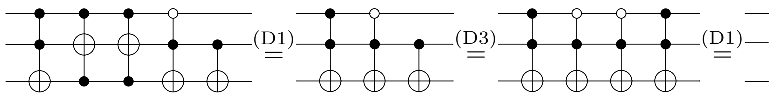
Given our new rewrite rules, it is possible to rewrite templates to the empty circuit. As an example we consider the following template which has been proposed in [11].



The arrows indicate all rewrite possibilities if only the classical moving rule can be applied, as it has been done in previous work. With the classical moving rule, the middle gates can be interchanged but also the first gate can be swapped with the last gate since the circuit represents the identity. However, it can easily be seen that the moving rules are not sufficient in order to remove gates from the circuit. In particular it is not possible to move the fifth gate next to the second, which is indeed possible using the proposed rewrite rules. For this purpose, we first apply Rule D3 to the fourth gate resulting in

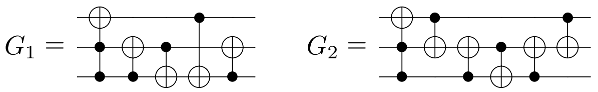


Note that the third and the fourth gate can be removed since they represent the identity and the fifth and sixth gate can be swapped since they have disjoint controls. Applying both rules allows to place the Toffoli gates with target on the second line next to each other:

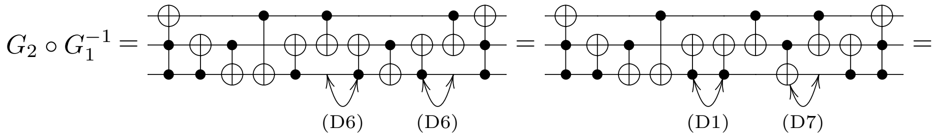


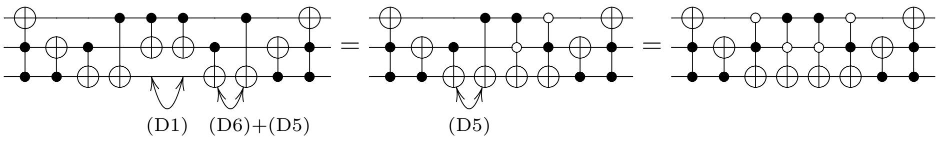
Following the previous example, we can apply rewrite rules in order to perform structural equivalence checking. Taking two circuits representing the functions G_1 and G_2, we can construct a new circuit that represents $G = G_1 \circ G_2^{-1}$ by appending the reverse of G_2 to G_1. If both circuits represent the same function, then G must represent the identity function and therefore it must be possible to rewrite G to the empty circuit.

We will illustrate this approach by means of an example in which we consider the circuits:



In order to rewrite $G_2 \circ G_1^{-1}$ to the empty circuit the following rewrite rules can be applied:





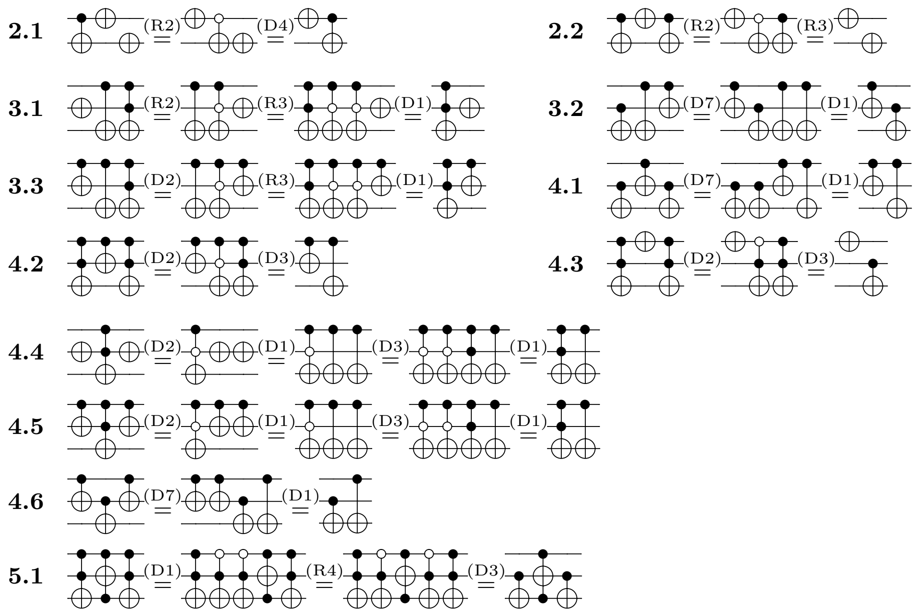


Fig. 2. Rewriting templates

Notice, that as a strategy we are bringing the combined circuit to the V-shape and finally apply the deletion rule starting from the inner gates towards the outer gates.

Figure 2 shows examples in which all Toffoli templates from [11] have been considered. The templates have been listed with a left-hand side and right-hand side circuit instead of giving the identity circuit explicitly. We applied our rewrite rules in order to write the left-hand side to the right-hand side. The identifiers have been taken from the original paper.

5 General Rewriting of Reversible Gates

All previous rules and derivations have, in essence, been specialised to a subset of the reversible logic gates, namely the general Toffoli gates. However, depending on the implementation technology (CMOS, quantum technology, *etc.*) other reversible gates are also of interest. In this section we will, therefore, show the first steps towards a general set of rules.

First, we need gate introduction / elimination rules similar to Rule R1:

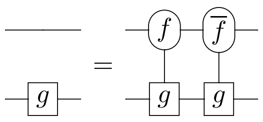
$$\text{———} = \text{—}\boxed{f}\text{—}\boxed{f^{-1}}\text{—}$$

where f can be any reversible logic function defined over any number of lines[1]. That this rule is true is obvious and if f is a self-inverse function (such as the

[1] As all lines can represent more than one bit-wire, we have, for aesthetic reasons, omitted the slash (/) on the lines.

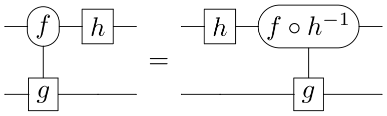
Not-gate), then $f = f^{-1}$. Notice also the similarity with the box-notation for grouping rules. If we remove the control, then the circuit in the box can be any reversible circuit.

This brings us to the second introduction / elimination rule, which is the general version of Rule R3:



where f can be any n-input Boolean function and g any reversible function. This exemplifies the *controlled-gate* structure that is used as the basis in all rules; we will call, for the first gate, f the control function of g. The semantics of the gates is simply that g is evaluated (updates its input wires) if and only if f evaluates to True.

An important part of the rules was the use of negative controls. Therefore, as a final example we show here the general version of Rule R2:



where the number of inputs to the control function f must equal the number of bit-wires that the reversible function h maps over.

This section has only shown a bit of the rules that is needed to describe the general system. Future work will provide the formalisation to general rules and show how other known reversible gates (*e.g.* the controlled-swap / Fredkin gate) relate to this.

6 Conclusion

In this paper, we have presented a rewrite system for reversible logic. We have illustrated our approach by means of Toffoli circuits but also illustrated how it can be generalised to be applied to all kind of reversible circuits. We have categorised the rules into basic ones and rules that can be derived from them. It turns out that the set of basic rules remains compact.

Since the rewrite rules can be applied to almost all possible gate combinations, an expensive matching step as it is required in similar methods such as template matching can be omitted. Instead, the rewrite rules require appropriate rewrite strategies in order to be applied efficiently, which we want to consider in future work, *e.g.* with rewrite strategies based on Boolean satisfiability (*cf.* with the template matching approach presented in [1]). Also to show completeness, in the sense that these rules can rewrite a circuit to all its equivalent circuits, is left for future work. Though not constructive, such a proof will also show that it is possible to rewrite a circuit into its minimal representation according to some metric.

Furthermore, we want to generalise the rewrite rules to arbitrary reversible circuits. Also the consideration of multiple-valued logic circuits, *e.g.* quantum circuits based on the NCV library, is an interesting target for future work.

Acknowledgement. The authors thank the *Danish Council for Strategic Research* for partially supporting this work through the *MicroPower* research project (`http://topps.diku.dk/micropower`).

References

1. Abdessaied, N., Soeken, M., Wille, R., Drechsler, R.: Exact template matching using Boolean satisfiability. In: Int'l. Symp. on Multiple-Valued Logic, ISMVL (2013)
2. Arabzadeh, M., Saeedi, M., Zamani, M.S.: Rule-based optimization of reversible circuits. In: Asia and South-Pacific Design Automation Conference, ASP-DAC, pp. 849–854 (2010)
3. Axelsen, H.B., Thomsen, M.K.: Garbage-free reversible integer multiplication with constants of the form $2^k \pm 2^l \pm 1$. In: Glück, R., Yokoyama, T. (eds.) RC 2012. LNCS, vol. 7581, pp. 171–182. Springer, Heidelberg (2013)
4. Baader, F., Nipkow, T.: Term rewriting and all that. Cambridge University Press, New York (1998)
5. Cuccaro, S.A., Draper, T.G., Kutin, S.A., Moulton, D.P.: A new quantum ripple-carry addition circuit arXiv:quant-ph/0410184v1 (2005)
6. Feynman, R.P.: Quantum mechanical computers. Optics News 11, 11–20 (1985)
7. Fredkin, E., Toffoli, T.: Conservative logic. International Journal of Theoretical Physics 21(3-4), 219–253 (1982)
8. Iwama, K., Kambayashi, Y., Yamashita, S.: Transformation rules for designing cnot-based quantum circuits. In: Design Automation Conference, DAC 2002, pp. 419–424. ACM (2002)
9. Landauer, R.: Irreversibility and heat generation in the computing process. IBM Journal of Research and Development 5(3), 183–191 (1961)
10. Maslov, D., Miller, D.M., Dueck, G.W.: Fredkin/Toffoli templates for reversible logic synthesis. In: Int'l Conf. on Computer Aided Design, ICCAD, pp. 256–261 (2003)
11. Miller, D.M., Maslov, D., Dueck, G.W.: A transformation based algorithm for reversible logic synthesis. In: Design Automation Conference, DAC, pp. 318–323 (2003)
12. Regergem, Y.V., Vos, A.D.: Young subgroups for reversible computers. Advances in Mathematics of Communications 2(2), 183–200 (2008)
13. Soeken, M., Wille, R., Hilken, C., Przigoda, N., Drechsler, R.: Synthesis of reversible circuits with minimal lines for large functions. In: Asia and South-Pacific Design Automation Conference, ASP-DAC, pp. 85–92 (2012)
14. Thomsen, M.K.: Describing and optimising reversible logic using a functional language. In: Gill, A., Hage, J. (eds.) IFL 2011. LNCS, vol. 7257, pp. 148–163. Springer, Heidelberg (2012)
15. Thomsen, M.K.: A functional language for describing reversible logic. In: Specification & Design Languages, FDL, pp. 135–142. IEEE (2012)

16. Thomsen, M.K., Glück, R., Axelsen, H.B.: Reversible arithmetic logic unit for quantum arithmetic. Journal of Physics A: Mathematical and Theoretical 43(38), 382002 (2010)
17. Toffoli, T.: Reversible computing. In: de Bakker, J.W., van Leeuwen, J. (eds.) ICALP 1980. LNCS, vol. 85, pp. 632–644. Springer (1980)
18. Van Rentergem, Y., De Vos, A., Storme, L.: Implementing an arbitrary reversible logic gate. Journal of Physics A: Mathematical and General (38), 3555–3577 (2005)
19. Vedral, V., Barenco, A., Ekert, A.: Quantum networks for elementary arithmetic operations. Physical Review A 54(1), 147–153 (1996)
20. Wille, R., Offermann, S., Drechsler, R.: SyReC: A programming language for synthesis of reversible circuits. In: Specification & Design Languages, FDL, pp. 1–6. IET (2010)
21. Yokoyama, T., Axelsen, H.B., Glück, R.: Principles of a reversible programming language. In: Conference on Computing Frontiers, CF, pp. 43–54. ACM Press (2008)