Appendix: Proofs of Lemmas, Propositions and Theorems

This is the appendix document for the paper "Engineering Grammar-based Type Checking for Graph Rewriting Languages", which consists of proofs for the propositions and theorems that appear in the paper.

Contents

A.1 Reverse execution and inversed rules (Proposition 1) 1
A.2 Graph type checking algorithm (Theorem 1–3) 1
A.2.1 Termination of graph type checking (Theorem 1) 1
A.2.2 Soundness of graph type checking (Theorem 2) 2
A.2.3 Completeness of graph type checking (Theorem 3) 2
A.3 Soundness of rule type checking algorithm (Theorem 4) 3
A.4 Functional atoms (Theorem 5) 4

A.1 Reverse execution and inversed rules (Proposition 1)

In Section III, we mentioned that we can perform reverse execution by applying inversed rules. Here we show the proof of Proposition 1, which guarantees this intuition is correct. Note that, hereinafter, for a set of functors F, Graph(F) stands for the set of every graph G s.t. Funct(G) $\subseteq F$.

Proposition 1. For LMNtal rule $r = T$:- U, $G' \xrightarrow{r} G \Leftrightarrow G \xrightarrow{r^{\text{inv}}} G'$

Proof. ($\Rightarrow$) We will prove by structural induction on the last-used reduction relation rule.

- Case (R1): We assume that $G' = P, Q,\ G = P', Q$ and $P \xrightarrow{r} P'$. By the induction hypothesis, $P' \xrightarrow{r^{\text{inv}}} P$. By (R1), $P', Q \xrightarrow{r^{\text{inv}}} P, Q$. Therefore, $G \xrightarrow{r^{\text{inv}}} G'$.
- Case (R3): We assume that $G' \equiv P,\ P' \equiv G,\ P \xrightarrow{r} P'$. By the induction hypothesis, $P' \xrightarrow{r^{\text{inv}}} P$. By (R3) and $G' \equiv P,\ P' \equiv G$, we obtain $G \xrightarrow{r^{\text{inv}}} G'$.
- Case (R6) is self-evident by (R6).

Hence $G' \xrightarrow{r} G \Rightarrow G \xrightarrow{r^{\text{inv}}} G'$.

($\Leftarrow$) follows from $(r^{\text{inv}})^{\text{inv}} = r$ and ($\Rightarrow$). $\square$

A.2 Graph type checking algorithm (Theorem 1–3)

In Section VI-B, we mentioned three theorems about the properties of the graph type checking algorithm: termination (Theorem 1), soundness (Theorem 2), and completeness (Theorem 3). Here we give a proof for each theorem with required lemmas.

A.2.1 Termination of graph type checking (Theorem 1)

Lemma A1. For a monotonic type $\tau = (t/m, P, N)$ with a weighting function w, and LMNtal graphs $G, G' \in$ Graph(Funct(τ)),

$$G' \xrightarrow{P} G \Rightarrow w(G) \geq w(G')$$

Proof. By $G' \xrightarrow{P} G$, let $p = \alpha$:- β be a rule s.t. $G' \xrightarrow{p} G$. We will show $w(G) \geq w(G')$ by structural induction on the last-used reduction relation rule.

- Case (R1): We assume that $G' = G'_1, G_2,\ G = G_1, G_2$ and $G'_1 \xrightarrow{p} G_1$. Then $w(G_1) \geq w(G'_1)$ by the induction hypothesis. We have $w(G') = w(G'_1) + w(G_2),\ w(G) = w(G_1) + w(G_2)$, therefore $w(G) \geq w(G')$ holds.
- Case (R3): We assume that $G'_1 \equiv G',\ G \equiv G_1,\ G'_1 \xrightarrow{p} G_1$. Then $w(G_1) \geq w(G'_1)$ by the induction hypothesis. We have $w(G_1) = w(G),\ w(G'_1) = w(G')$, therefore $w(G) \geq w(G')$ holds.

- Case (R6): We assume that $G' = \alpha,\ G = \beta$. Since the type $(t/m, P, N)$ is monotonic, we have $w(\alpha) \le w(\beta)$ i.e. $w(G) \ge w(G')$. □

Lemma A2. For a monotonic ShapeType $\tau = (t/m, P, N)$ with a weighting function w, a non-negative integer n, and a finite set of links L, the number of LMNtal graph G satisfying following formula is *finite* regarding structurally congruent graphs as the same.

$$G \in \text{Graph}(\text{Funct}(\tau)) \wedge w(G) = n \wedge \text{FLink}(G) = L$$

Proof. The number of functors occurring in G is finite and also the number of atoms (excluding non-global connectors) occurring in G is less than n because of $w(G) = n$. Since the number of graphs consisting of finite kinds of functors and finite atoms is finite, the number of G is finite. □

Lemma A3. For any LMNtal graph X, a monotonic ShapeType $(t/m, P, N)$, and a finite set of LMNtal graphs G, GGCheck$(X, (t/m, P, N, G))$ terminates.

Proof. Let the weighting function of type $(t/m, P, N)$ be w. Y given to the first argument of recursive call at line 5 of GGCheck satisfies $Y \xrightarrow{P}^{*} X$. By Lemma A1, $w(X) \ge w(Y)$. By this and Lemma A2, the number of possible Y is finite (regarding structurally congruent graphs are the same). Also, it is verified at line 3 of GGCheck that structurally congruent graphs cannot be given again to the argument of recursive calls, so that recursive calls occur finite times at most. Therefore GGCheck$(X, (t/m, P, N), G)$ terminates. □

Theorem 1 (Termination of graph type checking)**.** For any LMNtal graph X and a monotonic ShapeType τ, GCheck(X, τ) terminates.

Proof. This follows by Lemma A3 and the fact that Funct(X) and N are finite. □

A.2.2 Soundness of graph type checking (Theorem 2)

Lemma A4. For any LMNtal graph X, a ShapeType $(t/m, P, N)$, and a finite set of LMNtal graphs G, if GGCheck$(X, (t/m, P, N), G)$ returns **true**, $X \lhd t(L_1, \dots, L_m)$ holds.

Proof. We will show by induction on the maximum number n of times of recursive calls (i.e. the depth of recursion) of GGCheck.

- If $n = 0$, **true** is returned at line 2. Also, we have $X \equiv t(L_1, \dots, L_m)$. Then it is clear by the definition that $X \lhd t(L_1, \dots, L_m)$.
- If $n = k+1$ $(k \ge 0)$, **true** is returned at line 5 since recursive calls occur one or more times. We have $Y \xrightarrow{P} X$ by the line 3 and $Y \lhd t(L_1, \dots, L_m)$ by the induction hypothesis. Then $t(L_1, \dots, L_m) \xrightarrow{P}^{*} Y$ follows by the definition of production relations. By $Y \xrightarrow{P} X$, we have $Y \xrightarrow{P} X$, so that $t(L_1, \dots, L_m) \xrightarrow{P}^{*} X$. Therefore $X \lhd t(L_1, \dots, L_m)$ holds. □

Theorem 2 (Soundness of graph type checking)**.** For any LMNtal graph X and a ShapeType $(t/m, P, N)$, if GCheck$(X, (t/m, P, N))$ returns **true**, $X : t(L_1, \dots, L_m)$ holds.

Proof. GCheck$(X, (t/m, P, N))$ returns **true** only when $\exists f \in \text{Funct}(X).\ f \in N$ does not hold and then it just returns the returned value from GGCheck$(X, (t/m, P, N), \varnothing)$, so that GGCheck$(X, (t/m, P, N), \varnothing)$ returns **true**. By Lemma A4, we have $X \lhd t(L_1, \dots, L_m)$. Since $\neg\exists f \in \text{Funct}(X).\ f \in N$ holds, we have $\forall f \in \text{Funct}(X).\ f \notin N$. Therefore $\text{Funct}(X) \cap N = \varnothing$ holds. Hence we have $X : t(L_1, \dots, L_m)$. □

A.2.3 Completeness of graph type checking (Theorem 3)

Lemma A5. For an LMNtal graph X and a ShapeType $(t/m, P, N)$, if $X \lhd t(L_1, \dots, L_m)$, GGCheck$(X, (t/m, P, N), \varnothing)$ returns **true**.

Proof. By $X \lhd t(L_1, \dots, L_m)$, for certain $X_0, \dots, X_n$ $(n \ge 0)$, the following holds:

$$t(L_1, \dots, L_m) = X_n \xrightarrow{P} \dots \xrightarrow{P} X_1 \xrightarrow{P} X_0 = X$$

Note that X_i is not the start symbol for every i $(i < n)$ and $i \ne j \Rightarrow X_i \not\equiv X_j$ holds (i.e. no loops in the path). Consider the case when GGCheck$(Y_i, (t/m, P, N), G_i)$ is called for i $(i < n)$, Y_i s.t. $Y_i \equiv X_i$, and a certain G_i. Since X_i is not the start symbol, Y_i is also not the start symbol, so that the condition of the if statement at line 2 does not hold. Then we have $X_{i+1} \xrightarrow{P} Y_i$ by $X_{i+1} \xrightarrow{P} X_i$ and (R3).

- If $\nexists Y_{i+1} \in G_i.\ X_{i+1} \equiv Y_{i+1}$, the for-loop from the line 3 is executed for $Y \leftarrow X_{i+1}$, and then GGCheck$(X_{i+1}, (t/m, P, N), G_{i+1})$ is called for a certain G_{i+1} at line 5.
- If $\exists Y_{i+1} \in G_i.\ X_{i+1} \equiv Y_{i+1}$, GGCheck$(Y_{i+1}, (t/m, P, N), G_{i+1})$ has been called for a certain G_{i+1} elsewhere.

Therefore GGCheck$(Y_{i+1}, (t/m, P, N), G_{i+1})$ is called for Y_{i+1} s.t. $Y_{i+1} \equiv X_{i+1}$ and a certain G_{i+1} somewhere in the recursive calls.

From the above reasons, when GGCheck$(X, (t/m, P, N), \varnothing)$ is called, GGCheck$(Y_n, (t/m, P, N), G_n)$ is also called for Y_n s.t. $Y_n \equiv X_n$ and a certain G_n. This call returns **true** at line 2 because of $Y_n \equiv X_n = t(L_1, \ldots, L_m)$.

Therefore GGCheck$X, (t/m, P, N), \varnothing$ returns **true** since GGCheck(r)eturns **true** if one or more recursive calls in it return **true**. □

Theorem 3 (Completeness of graph type checking)**.** For any LMNtal graph X and a monotonic ShapeType $(t/m, P, N)$, if $X : t(L_1, \ldots, L_m)$ holds, GCheck$(X, (t/m, P, N))$ returns **true**.

Proof. By $X : t(L_1, \ldots, L_m)$, we have $X \lhd t(L_1, \ldots, L_m)$ and $\mathrm{Funct}(X) \cap N = \varnothing$. Therefore $\forall f \in Funct(X).\ f \notin X$, so that the condition of the if statement at line 2 of GCheck does not hold. Then GCheck$(X, (t/m, P, N))$ just returns the returned value from GGCheck$(X, (t/m, P, N), \varnothing)$. By $X \lhd t(L_1, \ldots, L_m)$ and Lemma A5, GGCheck$(X, (t/m, P, N), G)$ returns **true**. Therefore GCheck$(X, (t/m, P, N))$ returns **true**. □

A.3 Soundness of rule type checking algorithm (Theorem 4)

In Section VI-C, we mentioned a theorem about the soundness of the rule type checking algorithm (Theorem 4). Here we give a proof for this theorem with required lemmas.

Lemma A6. If $\alpha :\!-\ \beta \xrightarrow{r} \alpha' :\!-\ \beta'$ and $r \equiv r'$ holds, then we have $\alpha :\!-\ \beta \xrightarrow{r'} \alpha' :\!-\ \beta'$.

Proof. This follows by the rules of structural congruence and reduction relation (with additional rules (E11) and (E12)). □

Lemma 1. If $\alpha :\!-\ \beta$ is reducible and $\alpha :\!-\ \beta \xrightarrow{\mathcal{T}_P} \alpha' :\!-\ \beta'$, then $\alpha' :\!-\ \beta'$ is reducible.

Proof. By the definition of $\mathcal{T}_P$, we have

$$r \in P \ \wedge\ r' \equiv r \ \wedge\ \alpha :\!-\ \beta \xrightarrow{r'} \alpha' :\!-\ \beta'.$$

for certain rules r, r'. Hence we have $\alpha :\!-\ \beta \xrightarrow{r} \alpha' :\!-\ \beta'$ by Lemma A6. By this and the definition of rule transformation, we have

$$r = A :\!-\ B, C \ \wedge\ \alpha = B, L \ \wedge\ \alpha' = A, L \ \wedge\ \beta' = \beta, C$$

for certain graphs A, B, C, L. Now we have $\alpha' \xrightarrow{r} \alpha, C$ because of $\alpha' = A, L$ and $\alpha, C = B, L, C \equiv B, C, L$. In addition, $\alpha, C \xrightarrow{P}{}^{*} \beta, C\ (= \beta')$ follows by $\alpha \xrightarrow{P}{}^{*} \beta$. Thus we have $\alpha' \xrightarrow{P}{}^{*} \beta'$, that is, $\alpha' :\!-\ \beta'$ is reducible. □

Lemma A7. If $P, Q \equiv R, S$ holds, $P \equiv A_1, A_2,\ Q \equiv A_3, A_4,\ R \equiv A_1, A_3,\ S \equiv A_2, A_4$ holds for certain graphs A_1, A_2, A_3, A_4.

Proof. This follows by the rules of structural congruence. □

Lemma A8. If $P \xrightarrow{\alpha :- \beta} Q$ holds, there exists a graph C that satisfies $P \equiv C, \alpha$ and $Q \equiv C, \beta$.

Proof. This follows by the rules of structural congruence and reduction relation. □

Lemma A9. If $X \xrightarrow{p} Y \equiv \alpha, C\ (p = \alpha_p :\!-\ \beta_p \in P)$ holds, one of the followings holds:

- $\forall \beta.\ \exists \alpha', \beta', C_1, C_2.\ \alpha :\!-\ \beta \xrightarrow{\mathcal{T}_P} \alpha' :\!-\ \beta' \ \wedge\ C \equiv C_1, C_2 \wedge X \equiv \alpha', C_2 \wedge \beta' \equiv \beta, C_1$
- $\exists C'.\ X \equiv \alpha, C' \wedge C' \xrightarrow{p} C$

Proof. By $X \xrightarrow{p} Y$ and Lemma A8, there exists C_p s.t. $X \equiv \alpha_p, C_p,\ Y \equiv \beta_p, C_p$. Then we have $Y \equiv \beta_p, C_p \equiv \alpha, C$ and, by Lemma A7, there exist C_1, C_2, C_3, C_4 s.t. $C \equiv C_1, C_2,\ \alpha \equiv C_3, C_4,\ \beta_p \equiv C_1, C_3,\ C_p \equiv C_2, C_4$.

Case 1: $C_3 \not\equiv \mathbf{0}$

Let $\alpha' = \alpha_p, C_4,\ \beta' = \beta, C_1$. Then we have $\alpha' = \alpha_p, C_4 \xrightarrow{p} \beta_p, C_4 \equiv C_1, C_3, C_4 \equiv \alpha, C_1$. Now we have

$$C_3, C_4 \text{ :- } \beta \xrightarrow{\alpha_p \text{ :- } C_3, C_1} \alpha_p, C_4 \text{ :- } \beta, C_1$$

by the definition of rule transformation. Hence we have $\alpha \text{ :- } \beta \xrightarrow{\mathcal{T}_P} \alpha' \text{ :- } \beta'$. Besides, we have $X \equiv \alpha_p, C_p \equiv \alpha_p, C_2, C_4 \equiv \alpha', C_2$.

Case 2: $C_3 \equiv \mathbf{0}$

By $C_1 \equiv \beta_p$, we have $C \equiv \beta_p, C_2$. Therefore $\alpha_p, C_2 \xrightarrow{p} C$ holds. Here we consider C' s.t. $C' \equiv \alpha_p, C_2$, then $C' \xrightarrow{p} C$ holds. Besides, by $\alpha \equiv C_4$, we have $C_p \equiv C_2, \alpha$. Then we have $X \equiv \alpha_p, C_p \equiv \alpha_p, C_2, \alpha \equiv \alpha, C'$. □

Lemma 2. Given a type T with a production rule P, if the state space formed by the rule r and $\mathcal{T}_P$ satisfies $\neg\mathsf{start}\,\mathsf{W}\,\mathsf{red}$ holds, r preserves the type T.

Proof. We assume $G \lhd T$, $G \xrightarrow{r} G'$, and $\neg\mathsf{start}\,\mathsf{W}\,\mathsf{red}$, and we will prove $G' \lhd T$.

By $G \lhd T$, we have $\forall i < n.\ X_{i+1} \xrightarrow{p_i} X_i$ for a certain non-negative integer n, graphs $X_0, \ldots, X_n$ ($X_0 = G$, $X_n = T$), and $p_0, \ldots, p_{n-1} \in P$. By $G \xrightarrow{r} G'$, there exists $C \in \mathcal{G}(N \cup \Sigma)$ s.t. $G \equiv \alpha, C,\ G' \equiv \beta, C$ where $r = \alpha \text{ :- } \beta$.

Next, we will show that, if $X_i \equiv \alpha_i, C_i$ holds, there exist $\alpha_{i+1}, \beta_{i+1}, C_{i+1}$ such that:

$$\alpha_i \text{ :- } \beta_i \xrightarrow{\mathcal{T}}{}^{*} \alpha_{i+1} \text{ :- } \beta_{i+1} \ \wedge\ X_{i+1} \equiv \alpha_{i+1}, C_{i+1} \ \wedge\ \beta_{i+1}, C_{i+1} \xrightarrow{P}{}^{*} \beta_i, C_i$$

By $X_{i+1} \xrightarrow{p_i} X_i$ and Lemma A9, one of the following holds:

1. $\exists \alpha_{i+1}, \beta_{i+1}, C'_i, C_{i+1}.\ \alpha_i \text{ :- } \beta_i \xrightarrow{\mathcal{T}_P} \alpha_{i+1} \text{ :- } \beta_{i+1} \wedge C_i \equiv C'_i, C_{i+1} \wedge X_{i+1} \equiv \alpha_{i+1}, C_{i+1} \wedge \beta_{i+1} \equiv \beta_i, C'_i$
2. $\exists C_{i+1}.\ X_{i+1} \equiv \alpha_i, C_{i+1} \wedge C_{i+1} \xrightarrow{p_i} C_i$

If 1. holds, it is obvious since we have $\beta_{i+1}, C_{i+1} \equiv \beta_i, C'_i, C_{i+1} \equiv \beta_i, C_i$. On the other hand, if 2. holds, it is obvious when we consider $\alpha_{i+1} = \alpha_i,\ \beta_{i+1} = \beta_i$.

Hence there exist α_n, β_n, C_n s.t. $\alpha \text{ :- } \beta \xrightarrow{\mathcal{T}_P}{}^{*} \alpha_n \text{ :- } \beta_n$, $T \equiv \alpha_n, C_n$, and $\beta_n, C_n \xrightarrow{P}{}^{*} \beta, C$. Since T consists only of one atom of the start symbol (with no self loops), we have $T \equiv \alpha_n$. Thus $\alpha_n \text{ :- } \beta_n$ is a start rule.

By $r \xrightarrow{\mathcal{T}_P}{}^{*} \alpha_n \text{ :- } \beta_n$, $\neg\mathsf{start}\,\mathsf{W}\,\mathsf{red}$, and Lemma 1, we also have that $\alpha_n \text{ :- } \beta_n$ is reducible. Therefore we have $\alpha_n \xrightarrow{P}{}^{*} \beta_n$. Thus we have $T \equiv \alpha_n \xrightarrow{P}{}^{*} \beta_n \xrightarrow{P}{}^{*} \beta, C \equiv G'$, that is, $G' \lhd T$. □

Theorem 4 (Soundness of rule type checking)**.** For an LMNtal rule r, a ShapeType $(t/m, P, N)$, and a sequence of links $L_1, \ldots, L_m$, if RCheck$(\alpha \text{ :- } \beta, (t/m, P, N))$ returns **true**, the following formula (the rule preserving property) holds:

$$\forall G, G'.\quad G : t(L_1, \ldots, L_m) \ \wedge\ G \xrightarrow{\alpha \text{ :- } \beta} G' \Rightarrow G' : t(L_1, \ldots, L_m)$$

Proof. Since RCheck$(\alpha \text{ :- } \beta, (t/m, P, N))$ returns **true**, on all the paths from the target rule to the start symbol, there exists a state $L \text{ :- } R$ such that Reduce$(R, L, P, \varnothing)$ returns **true**. Therefore we have $\neg\mathsf{start}\,\mathsf{W}\,\mathsf{red}$, and the rule preserving property holds by Lemma 2. □

A.4 Functional atoms (Theorem 5)

In Section VII, we defined Functional Property (Def. 18) to formalize functional atoms that are commonly used design pattern in LMNtal. Here we give a proof of Theorem 5, which states that we can ensure that a given functor has the functional property using the rule type checking.

Theorem 5. For a set of production rules $P = P_T \cup P_{t_1} \cup \cdots \cup P_{t_n} \cup \{T \text{ :- } F, t_1, \ldots, t_n\}$, and a set of nonterminal symbols $N = N_T \cup N_{t_1} \cup \cdots \cup N_{t_n}$, if every rule $r \in R$ preserves type (S_T, P, N), $t_1, \ldots, t_n \vdash_R F : T$ holds.

Proof. $F, G_{t_1}, \ldots, G_{t_n}$ has the type (S_T, P, N) because P includes the production rule $T \text{ :- } F, t_1, \ldots, t_n$ and T is the start symbol. Let G be a graph s.t. $F, G_{t_1}, \ldots, G_{t_n} \xrightarrow{R}{}^{*} G$. Then G has the type (S_T, P, N) because every rule $r \in R$ preserves the type. Here we assume that G contains no f/m atoms. By $G : (S_T, P, N)$, there exists a production path s.t. $S_T \xrightarrow{P}{}^{*} G$. Since G contains no f/m atoms, the production rule $T \text{ :- } F, t_1, \ldots, t_n$ has not been applied in the production path. Also the nonterminal symbols of the types $t_1, \ldots, t_n$ do not appear in the production path because they can appear only after the production rule T :- F, t_1, ..., t_n is applied. Therefore the production rules of the types $t_1, \ldots, t_n$ have not been applied in the production path, so that the nonterminal symbols $N_{t_1}, \ldots, N_{t_n}$ and the production rules $P_{t_1}, \ldots, P_{t_n}$ are redundant in the production path. Hence we have $G : (S_T, P_T, N_T) = T$. By the definition of functional property, $t_1, \ldots, t_n \vdash_R F : T$ holds. $\square$