## 0.1 Congruence Path Refinement

In this section we present an algorithm that uses an existing congruence relation and refines it to the point where equivalent states can be "merged".

**Definition 0.1.1.** Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA and let $\equiv \subseteq Q \times Q$ be an equivalence relation on the state set. For every equivalence class $\kappa \subseteq Q$, let $r_\kappa \in \kappa$ be an arbitrary representative of that class. For a DPA $\mathcal{A}' = (Q', \Sigma, q_0', \delta', c')$, we say that $\mathcal{A}'$ is a *representative merge of* $\mathcal{A}$ *w.r.t.* $\equiv$ if it satisfies the following:

- $Q' = \{r_{[q]_\equiv} \subseteq Q \mid q \in Q\}$
- $q_0' = r_{[q_0]_\equiv}$
- For all $q \in Q'$ and $a \in \Sigma$: $\delta'(q, a) = r_{[\delta(q,a)]_\equiv}$
- $c' = c \restriction_{Q'}$

**Definition 0.1.2.** Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA and let $R \subseteq Q \times Q$ be a congruence relation on the state space. Let $\lambda \subseteq Q$ be an equivalence class of $R$. We define $L_{\lambda\leftarrow}$ as the set of words $w$ such that for any $u \sqsubseteq w$, $(\delta(p, u), q) \in R$ iff $u \in \{\varepsilon, w\}$. In other words, the set contains all minimal words by which the automaton moves from $\lambda$ to $\lambda$ again.

Let $f_{\mathrm{PR}} : 2^{\lambda\times\lambda} \to 2^{\lambda\times\lambda}$ be a function such that $(p, q) \in f(X)$ iff for all $w \in L_{\lambda\leftarrow}$, $(\delta^*(p, w), \delta^*(q, w)) \in X$. Then let $X_0 \subseteq \lambda \times \lambda$ such that $(p, q) \in X_0$ iff for all $w \in L_{\lambda\leftarrow}$, $\min\{c(\delta^*(p, u)) \mid u \sqsubseteq w\} = \min\{c(\delta^*(q, u)) \mid u \sqsubseteq w\}$, i.e. the minimal priority when moving from $p$ or $q$ to $\lambda$ again is the same.

Using both, we set $X_{i+1} = f_{\mathrm{PR}}(X_i)$. $f_{\mathrm{PR}}$ is monotone w.r.t. $\subseteq$, so there is an $X_n = X_{n+1}$ by Kleene's fixed point theorem. We define the *path refinement of* $\lambda$, called $\equiv^\lambda_{\mathrm{PR}}$, as

- For $p \in Q \setminus \lambda$, $p \equiv^\lambda_{\mathrm{PR}} q$ iff $p = q$.
- For $p, q \in \lambda$, $p \equiv^\lambda_{\mathrm{PR}} q$ iff $(p, q) \in X_n$.

**Theorem 0.1.1.** *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA and let $R \subseteq Q \times Q$ be a congruence relation that implies language equivalence. Let $\mathcal{A}'$ be a representative merge of $\mathcal{A}$ w.r.t. $\equiv^\lambda_{PR}$ for some equivalence class $\lambda$ of $R$ such that each representative $r_\kappa$ is chosen to have minimal priority. Then $L(\mathcal{A}) = L(\mathcal{A}')$.*

*Proof.* Let $\alpha \in \Sigma^\omega$ be a word with runs $\rho \in Q^\omega$ and $\rho' \in (Q')^\omega$ of $\mathcal{A}$ and $\mathcal{A}'$ respectively. Let $k_0, \cdots \in \mathbb{N}$ be exactly those positions (in order) at which $\rho$ reaches $\lambda$, and analogously $k_0', \ldots$ for $\rho'$.

**Claim 1**: For every $i$, $k_i = k_i'$ and $\rho(k_i) \equiv^\lambda_{\mathrm{PR}} \rho'(k_i)$.
For all $j < k_0$, we know that $\rho(j) = \rho'(j)$, as no redirected edge is taken. Thus, $\rho'(k_0) = r_{[\rho(k_0)]_{\equiv^\lambda_{\mathrm{PR}}}} \equiv^\lambda_{\mathrm{PR}} = \rho(k_0)$.

Now assume that the claim holds for all $i \leq n$. By definition, $w = \alpha[k_n, k_{n+1}] \in L_{\lambda\leftarrow}$ and therefore $\rho(k_{n+1}) = \delta^*(\rho(k_n), w) \equiv^\lambda_{\mathrm{PR}} \delta^*(\rho'(k_n), w) = \rho'(k_{n+1})$.

**Claim 2**: If $\lambda$ only occurs finitely often in $\rho$ and $\rho'$, then $\rho$ is accepting iff $\rho'$ is accepting.

Let $k_n \in \mathbb{N}$ be the last position at which $\rho(k_n)$ and $\rho'(k_n)$ are in $\lambda$. From this point on, $\rho'[k_n, \omega]$ is also a valid run of $\mathcal{A}$ on $\alpha[k_n, \omega]$. $\rho(k_n), \rho'(k_n) \in \lambda$, so $(\rho(k_n), \rho'(k_n)) \in R$. As $R$ implies language

equivalence, reading $\alpha[k_n, \omega]$ from either state in $\mathcal{A}$ leads to the same acceptance status. This also means that $\rho'(k_n)$ has the same acceptance status as $\rho(k_n)$.

**Claim 3**: If $\lambda$ occurs infinitely often in $\rho$ and $\rho'$, then $\rho$ is accepting iff $\rho'$ is accepting.

We show that for all $i$, $\min \mathrm{Occ}(c(\rho[k_i, k_{i+1}+1]))$ and $\min \mathrm{Occ}(c'(\rho'[k_i, k_{i+1}+1]))$ are the same. The claim then follows immediately.

Directly observe that $c'(\rho'[k_i, k_{i+1}+1]) = c(\rho'[k_i, k_{i+1}+1])$ and that $\min \mathrm{Occ}(c(\rho[k_i, k_{i+1}+1])) = \min \mathrm{Occ}(c(\rho'[k_i, k_{i+1}] \cdot \delta(\rho'(k_{i+1}-1), \alpha(k_{i+1}))))$ because $\rho(k_i) \equiv^{\lambda}_{\mathrm{PR}} \rho'(k_i)$.

Now $\mathrm{Occ}(c(\rho[k_i, k_{i+1}+1])) = \mathrm{Occ}(c(\rho[k_i, k_{i+1}])) \cup \{c(\rho(k_{i+1}))\}$ and $\mathrm{Occ}(c(\rho'[k_i, k_{i+1}] \cdot \delta(\rho'(k_{i+1}-1), \alpha(k_{i+1}))) = \mathrm{Occ}(c(\rho[k_i, k_{i+1}])) \cup \{c(\delta(\rho'(k_{i+1}-1), \alpha(k_{i+1})))\}$.

The rest of the claim follows because $c(\rho'(k_i)) = c(\rho'(k_{i+1})) \leq \delta(\rho'(k_{i+1}, \alpha(k_{i+1}))$. □

### 0.1.1 Algorithmic Definition

The definition of path refinement that we introduced is useful for the proofs of correctness. It however does not provide one with a way to actually compute the relation. That is why we now provide an alternative definition that yields the same results but is more algorithmic in nature.

**Definition 0.1.3.** Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA and let $R \subseteq Q \times Q$ be a congruence relation. For each equivalence class $\lambda$ of $R$, we define the *path refinement automaton* $\mathcal{G}^{R,\lambda}_{\mathrm{PR}}(p, q) = (Q_{\mathrm{PR}}, \Sigma, q^{p,q}_{0,\mathrm{PR}}, \delta^{\lambda}_{\mathrm{PR}}, F_{\mathrm{PR}})$, which is a DFA.

- $Q_{\mathrm{PR}} = (Q \times Q \times c(Q) \times \{<, >, =\}) \cup \{\bot\}$
- $q^{p,q}_{0,\mathrm{PR}} = (p, q, \eta_k(c(p), c(q), \checkmark), \eta_x(c(p), c(q), \checkmark, =))$
- $\delta^{\lambda}_{\mathrm{PR}}((p, q, k, x), a) = \begin{cases} (p', q', \eta_k(c(p'), c(q'), k), \eta_x(c(p'), c(q'), k, x)) & \text{if } p' \notin \lambda \\ q^{p',q'}_{0,\mathrm{PR}} & \text{if } p' \in \lambda \text{ and } (\eta_x(c(p'), c(q'), k, x) == ) \\ \bot & \text{else} \end{cases}$

  where $p' = \delta(p, a)$ and $q' = \delta(q, a)$.

  $\eta_k(k_p, k_q, k) = \min_{\leq_\checkmark}\{k_p, k_q, k\}$

  $\eta_x(k_p, k_q, k, x) = \begin{cases} < & \text{if } (k_p <_\checkmark k_q \text{ and } k_p <_\checkmark k) \text{ or } (k < k_q \text{ and } (x = <)) \\ > & \text{if } (k_p >_\checkmark k_q \text{ and } k >_\checkmark k_q) \text{ or } (k_p > k \text{ and } (x = >)) \\ = & \text{else} \end{cases}$
- $F_{\mathrm{PR}} = Q_{\mathrm{PR}} \setminus \{\bot\}$

**Lemma 0.1.2.** *Let $\mathcal{A}$ be a DPA with a congruence relation $R$. Let $\lambda$ be an equivalence class of $R$, $p, q \in \lambda$, and $w \in L_{\lambda \to \lambda}$. For every $v \sqsubset w$ and $\oplus \in \{<, >, =\}$, the fourth component of $(\delta^{\lambda}_{PR})^*(q_{0,PR}, v)$ is $\oplus$ if and only if $\min\{c(\delta^*(p, u)) \mid u \sqsubseteq v\} \oplus \min\{c(\delta^*(q, u)) \mid u \sqsubseteq v\}$.*

The proof of this Lemma is a very formal analysis of every case in the relations between the different priorities that occur and making sure that the definition of $\eta_x$ covers these correctly. No great insight is gained, which is why we omit the proof at this point.

**Theorem 0.1.3.** *Let $\mathcal{A}$ be a DPA with a congruence relation $R$. Let $\lambda$ be an equivalence class of $R$ and $p, q \in \lambda$. Then $p \equiv^{R}_{PR} q$ iff $L(\mathcal{G}^{R,\lambda}_{PR}(p, q)) = \Sigma^*$.*

*Proof.* **If** Let $p \not\equiv^R_{\mathrm{PR}} q$. Similarly to the proof of Lemma **??**, we use the inductive definition of $R_\kappa \subseteq \equiv^R_{\mathrm{PR}}$ using $f$ and the sets $X_i$ here. Let $m$ be the smallest index at which $(p,q) \notin X_m$. Let $\rho = (p_i, q_i, k_i, x_i)_{0 \le i \le |w|}$ be the run of $\mathcal{G}^{R,\lambda}_{\mathrm{PR}}(p,q)$ on $w$. We prove that $\rho(|w|) = \bot$ and therefore $\rho$ is not accepting by induction on $m$.

If $m = 0$, then $(p,q) \notin Y_\lambda$, meaning that there is a word $w$ such that $\min\{c(\delta^*(p,u)) \mid u \sqsubset w\} \neq \min\{c(\delta^*(q,u)) \mid u \sqsubset w\}$. Without loss of generality, assume $\min\{c(\delta^*(p,u)) \mid u \sqsubset w\} < \min\{c(\delta^*(q,u)) \mid u \sqsubset w\}$. By Lemma 0.1.2, $x_{|w|-1} = <$. Furthermore, $\delta(p_{|w|-1}, w_{|w|-1}) \in \lambda$, as $w \in L_{\lambda\to\lambda}$. Thus, $\rho(|w|) = \bot$ and the run is rejecting.

Now consider $m + 1 > 1$. Since $(p,q) \in X_m \setminus f(X_m)$, there must be a word $w \in L_{\lambda\to\lambda}$ such that $(p',q') \notin X_m$, where $p' = \delta^*(p,w)$ and $q' = \delta^*(q,w)$. As $R_\kappa \subseteq X_m$, $(p',q') \notin R_\kappa$ and therefore $p' \not\equiv^R_{\mathrm{PR}} q'$. By induction, $w \notin L(\mathcal{G}^{R,\lambda}_{\mathrm{PR}}(p',q'))$; since that run is a suffix of $\rho$, $\rho$ itself is also a rejecting run.

**Only If** Let $L(\mathcal{G}^{R,\lambda}_{\mathrm{PR}}(p,q)) \neq \Sigma^*$. Since $\varepsilon$ is always accepted, there is a word $w \in \Sigma^+ \setminus L(\mathcal{G}^{R,\lambda}_{\mathrm{PR}}(p,q))$, meaning that $\delta^*_{\mathrm{PR}}(q_{0,\mathrm{PR}}, w) = \bot$. Split $w$ into sub-words $w = u_1 \cdots u_m$ such that $u_1, \ldots, u_m \in L_{\lambda\to\lambda}$. Note that this partition is unique. We show $p \not\equiv^R_{\mathrm{PR}} q$ by induction on $m$. Let $\rho = (p_i, q_i, k_i, x_i)_{0 \le i < |w|}$ be the run of $\mathcal{G}^{R,\lambda}_{\mathrm{PR}}(p,q)$ on $w$.

If $m = 1$, then $w \in L_{\lambda\to\lambda}$. Since $\rho(|w|) = \bot$, it must be true that $x_{|w|-1} \neq =$. Without loss of generality, assume $x_{|w|-1} = <$. By Lemma 0.1.2, $\min\{c(\delta^*(p,u)) \mid u \sqsubseteq w\} < \min\{c(\delta^*(q,u)) \mid u \sqsubseteq w\}$. Therefore, $p \not\equiv^R_{\mathrm{PR}} q$.

Now consider $m + 1 > 1$. Let $p' = \delta^*(p, u_1)$ and $q' = \delta^*(q, u_1)$. By induction on the word $u_2 \cdots u_m$, $p' \not\equiv^R_{\mathrm{PR}} q'$. Since $u_1 \in L_{\lambda\to\lambda}$, that also means $p \not\equiv^R_{\mathrm{PR}} q$. □

The differences between different $\mathcal{G}^{R,\lambda}_{\mathrm{PR}}$ for different $\lambda$ are minor and the question whether the accepted language is universal boils down to a simple question of reachability. Thus, $\equiv^R_{\mathrm{PR}}$ can be computed in $\mathcal{O}(|\mathcal{G}^{R,\lambda}_{\mathrm{PR}}|)$ which is $\mathcal{O}(|Q|^2 \cdot |c(Q)|)$.

### 0.1.2 Alternative Algorithmic Definition

The computation presented in the previous section was a straight-forward description of $\equiv^\lambda_{\mathrm{PR}}$ in an algorithmic way. We can reduce the complexity of that computation by taking a more indirect route, as we will see now.

**Definition 0.1.4.** Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ be a DPA. Let $R$ be a congruence relation on $Q$ and let $\lambda \subseteq Q$ be an equivalence class of $R$. We define a deterministic transition structure $\mathcal{A}^\lambda_{\mathrm{visit}} = (Q^\lambda_{\mathrm{visit}}, \Sigma, \delta^\lambda_{\mathrm{visit}})$ as follows:

- $Q^\lambda_{\mathrm{visit}} = ((Q \setminus \lambda) \times c(Q)) \cup (\lambda \times c(Q) \times c(Q))$
  These states "simulate" $\mathcal{A}$ and use the second component to track the minimal priority that was seen since a last visit to $\lambda$. The states in $\lambda$ itself also have a third component that is used to distinguish their classes, as is explained below.
- For $q \in Q \setminus \lambda$: $\delta^\lambda_{\mathrm{visit}}((q,k),a) = \begin{cases} (q', \min\{k, c(q')\}) & \text{if } q' \notin \lambda \\ (q', c(q'), \min\{k, c(q')\}) & \text{if } q' \in \lambda \end{cases}$.
  For $q \in \lambda$: $\delta^\lambda_{\mathrm{visit}}((q,k,k'),a) = (q', \min\{k, c(q')\})$.
  Where $q' = \delta(q,a)$.

**Definition 0.1.5.** Consider $\mathcal{A}^{\lambda}_{\text{visit}}$ of a DPA $\mathcal{A}$ and a congruence relation $R$. We define an equivalence relation $V \subseteq Q^{\lambda}_{\text{visit}} \times Q^{\lambda}_{\text{visit}}$ as:

- For every $p, q \in Q \setminus \lambda$ and $l, k \in c(Q)$, $((p, l), (q, k)) \in V$.
- For every $p, q \in \lambda$ and $l, k \in c(Q)$, $((p, l, l'), (q, k, k')) \in V$ iff $l' = k'$.

The Moore-refinement of $V$ is then called $V_M$.

**Lemma 0.1.4.** *Let $\mathcal{A}$, $R$, and $\lambda$ be as before. For $k \in c(Q)$ and $q \in Q$, we call $\iota^k_q = (q, c(q), k)$. Then for all $p, q \in \lambda$, and $l, k \in c(Q)$, $(\iota^k_p, \iota^k_q) \in V_M$ if and only if $(\iota^l_p, \iota^l_q) \in V_M$.*

*Proof.* □

**Theorem 0.1.5.** *Let $\mathcal{A}$, $R$, and $\lambda$ be as before. For $k \in c(Q)$ and $q \in Q$, we call $\iota^k_q = (q, c(q), k)$. Then for all $p, q \in \lambda$, and $k \in c(Q)$, we have $p \equiv^{\lambda}_{PR} q$ iff $(\iota^k_p, \iota^k_q) \in V_M$.*

*Proof.* We have $(p_0, q_0) \in V_M$ if and only if for all $w \in \Sigma^*$: $(\delta^*(p_0, w), \delta^*(q_0, w)) \in V$. Thus, we want to show that his property holds for all $w$ iff $p \equiv^{\lambda}_{\text{PR}} q$.

**If** Assume there is a $w$ such that $(\delta^*(p_0, w), \delta^*(q_0, w)) \notin V$.

**Only If** Assume $p \not\equiv^{\lambda}_{\text{PR}} q$. Let $f_{\text{PR}}$ and $(X_i)_i$ be the function and sets used in the construction of the path refinement. Let $n$ be minimal s.t. $(p, q) \notin X_n$. We use induction on $n$ to prove the claim.

If $n = 0$,

For $n + 1 > 0$, there is a $v \in L_{\lambda \hookleftarrow}$ such that $(\delta^*(p, v), \delta^*(q, v)) \notin X_n$. By induction, there is a word $u$ such that $(\delta^*(\iota^l_{\delta^*(p,v)}, u), \delta^*(\iota^{l'}_{\delta^*(q,v)}, u)) \notin V$.

If $u = \varepsilon$, then $(\iota^l_{\delta^*(p,v)}, \iota^{l'}_{\delta^*(q,v)}) \notin V$.

Otherwise, $u \neq \varepsilon$. □

The automaton has size $|\mathcal{A}^{\lambda}_{\text{visit}}| \in \mathcal{O}(|Q| \cdot |c(Q)|^2)$ and the computation of $V_M$ brings the runtime up to $\mathcal{O}(|\mathcal{A}^{\lambda}_{\text{visit}}| \cdot \log |\mathcal{A}^{\lambda}_{\text{visit}}|)$.