Two-Way Automata in Coq

Christian Doczkal and Gert Smolka

Interactive Theorem Proving, Nancy, France, August 24, 2016

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Motivation

- Myhill-Nerode in Isabelle/HOL based on regular expressions (Wu, Zhang, Urban '11)
- Various automata formalizations in dependent type theory:
  - Myhill-Nerode based on automata in Nuprl (Constable '00)
  - Coq tactic for deciding RE equivalence (Coquand Siles '11)
  - Coq tactic for deciding Kleene algebras (Braibant Pous '12)
- Student Project: Elegant formalization of automata/Myhill-Nerode based on Ssreflect's finite types.
- Equivalence of DFAs, NFAs, and REs and constructive variant of Myhill-Nerode in Coq (Doczkal Kaiser Smolka '13)
- Today: Reduction from Two-Way automata to DFAs based on constructive Myhill-Nerode theorem formalized constructively in Coq.

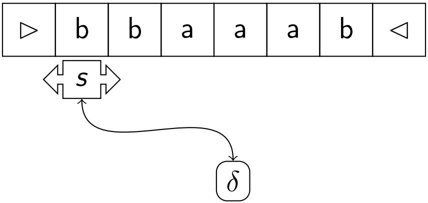
Two-Way Automata

- Another representation of regular languages
- Essentially: "Read-only Turing machines without memory"
- Introduced together with one-way automata (Rabin Scott '59)
- Reductions to DFAs in (Rabin Scott '59) and (Shepherdson '59)
- Reduction from 2NFAs to NFAs for complement language (Vardi '89)
- Recent Survey Paper by (Pighizzini '13):
  "Two-Way Auotmata: Old and Recent Results"

Two-Way Automata

2NFA $M = (Q, s, F, \delta)$ where

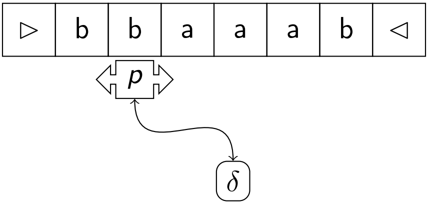
- Q is a finite type of *states*
- $s : Q$ is the *starting state*
- $F : 2^Q$ is the set of *final states*
- $\delta : Q \to \Sigma \uplus \{\triangleright, \triangleleft\} \to 2^{Q \times \{\mathsf{L},\mathsf{R}\}}$ is the transition function.



Two-Way Automata

2NFA $M = (Q, s, F, \delta)$ where

- Q is a finite type of *states*
- $s : Q$ is the *starting state*
- $F : 2^Q$ is the set of *final states*
- $\delta : Q \to \Sigma \uplus \{\triangleright, \triangleleft\} \to 2^{Q \times \{L,R\}}$ is the transition function.

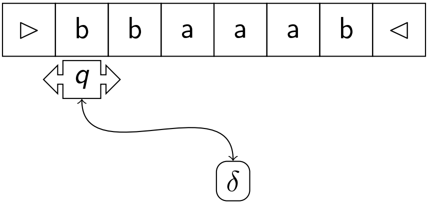


Two-Way Automata

SAARLAND UNIVERSITY
COMPUTER SCIENCE

2NFA $M = (Q, s, F, \delta)$ where

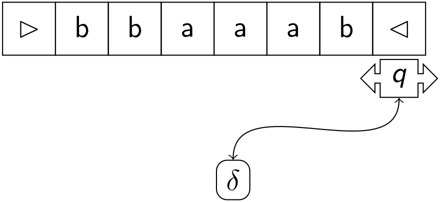
- Q is a finite type of *states*
- $s : Q$ is the *starting state*
- $F : 2^Q$ is the set of *final states*
- $\delta : Q \to \Sigma \uplus \{\triangleright, \triangleleft\} \to 2^{Q \times \{\mathsf{L},\mathsf{R}\}}$ is the transition function.



Language of a Two-Way Automaton

- Configurations on word x: $C_x := Q \times \{0, \dots, |x| + 1\}$.
- Step relation on x: $(p, i) \underset{x}{\rightarrow} (q, j) : C_x \rightarrow C_x \rightarrow \mathbb{B}$.

$$\mathcal{L}(M) := \{\, x \in \Sigma^* \mid \exists q \in F.\, (s, 1) \underset{x}{\rightarrow}^{*} (q, |x| + 1) \,\}$$



1. Language membership is (obviously) decidable
2. Main Result: 2DFAs and 2NFAs accept exactly the regular languages
   (M is a deterministic (a 2DFA) if $\underset{x}{\rightarrow}$ is functional for all x.)

Two-Way vs. One-Way

$$I_n := (a+b)^* a (a+b)^{n-1}$$

automata model	size of minimal automaton
DFA	$\mathcal{O}(2^n)$
NFA	$\mathcal{O}(n)$
2DFA	$\mathcal{O}(n)$

- Cost (in the number of states) of simulating 2DFAs with DFAs is at least exponential.
- Conjecture (Sakoda & Sipser '78): The cost of simulating NFAs using 2DFAs is exponential.

Main Results

SAARLAND UNIVERSITY
COMPUTER SCIENCE

1. Vardi '89:
   n-state 2NFA for $L \rightsquigarrow$ NFA for $\overline{L}$ with at most 2^{2n} states

2. Shepherdson '59:
   n-state 2DFA for $L \rightsquigarrow$ DFA for L with at most $(n+1)^{(n+1)}$ states

3. Shepherdson '59 $\cup$ folklore $\cup$ Vardi '89:
   n-state 2NFA for $L \rightsquigarrow$ DFA for L with at most 2^{n^2+n} states

Main Results

SAARLAND UNIVERSITY
COMPUTER SCIENCE

1. Vardi '89:
   n-state 2NFA for $L \rightsquigarrow$ NFA for $\overline{L}$ with at most 2^{2n} states

2. Shepherdson '59:
   n-state 2DFA for $L \rightsquigarrow$ DFA for L with at most $(n+1)^{(n+1)}$ states

3. Shepherdson '59 ∪ folklore ∪ Vardi '89:
   n-state 2NFA for $L \rightsquigarrow$ DFA for L with at most 2^{n^2+n} states

Vardi Construction

Input: 2NFA $M = (Q, s, F, \delta)$
Output: NFA accepting $\overline{\mathcal{L}(M)}$

▷	b	b	a	a	a	b	◁

(extended) word in $\overline{\mathcal{L}(M)}$

C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7

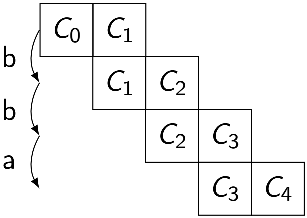
negative certificate

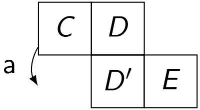
Definition (Negative Certificate for x)

1. $s \in C_1$
2. If $p \in C_i$ and $(p, i) \underset{x}{\rightarrow} (q, j)$, then $q \in C_j$
3. $F \cap C_{|x|+1} = \emptyset$.

Construct NFA N that accepts words that have negative certificates

Vardi Construction





1. $D = D'$
2. If $p \in D$ and $(q, \mathrm{L}) \in \delta\, p\, a$, then $q \in C$
3. If $p \in D$ and $(q, \mathrm{R}) \in \delta\, p\, a$, then $q \in E$

$N := (Q', S', F', \delta')$ where

$$Q' := 2^Q \times 2^Q$$

Vardi Construction

Input: 2NFA $M = (Q, s, F, \delta)$
Output: NFA accepting $\overline{\mathcal{L}(M)}$

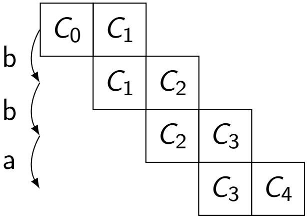
$\triangleright$	b	b	a	a	a	b	$\triangleleft$	(extended) word in $\overline{\mathcal{L}(M)}$
C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	negative certificate

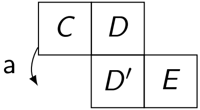
Definition (Negative Certificate for x)

1. $s \in C_1$
2. If $p \in C_i$ and $(p, i) \underset{x}{\rightarrow} (q, j)$, then $q \in C_j$
3. $F \cap C_{|x|+1} = \emptyset$.

Construct NFA N that accepts words that have negative certificates

Vardi Construction





1. $D = D'$
2. If $p \in D$ and $(q, \mathsf{L}) \in \delta\, p\, a$, then $q \in C$
3. If $p \in D$ and $(q, \mathsf{R}) \in \delta\, p\, a$, then $q \in E$

$N \; := (Q', S', F', \delta')$ where

$$Q' \; := \; 2^Q \times 2^Q$$

$$S' \; := \{\, (C_0, C_1) \mid s \in C_1, \forall pq.\, p \in C_0 \wedge (q, \mathsf{R}) \in \delta\, p \triangleright \to q \in C_1 \,\}$$

$$F' \; := \{\, (C, D) \mid F \cap D = \emptyset, \forall pq.\, p \in D \wedge (q, \mathsf{L}) \in \delta\, p \triangleleft \to q \in C \,\}$$

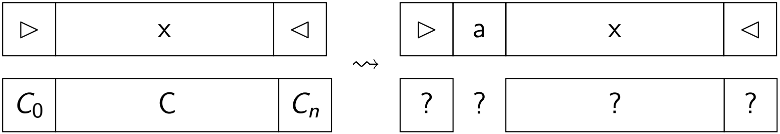
The Need for Runs

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Lemma

$x \in \mathcal{L}(N)$ iff there exists a negative certificate for x.

- Usually: generalize to arbitrary states of N and use induction on x
- direct inductive proof would require a nontrivial generalization



- Formalized proof employs an explicit notion of run ($\approx 1/3$ of proof)
- Straightforward but tedious calculation

Vardi Result

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Theorem

For every n-state 2NFA M one can construct an NFA accepting $\overline{\mathcal{L}(M)}$ that has at most 2^{2n} states.

(recall: $Q' := 2^Q \times 2^Q$ and $|2^Q| = 2^{|Q|}$ due to extensional representation)

Corollary

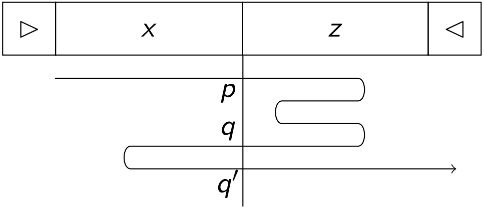
For every n-state 2NFA M there exists a DFA accepting $\mathcal{L}(M)$ that has at most $2^{2^{2n}}$ states.

Shepherdson Construction

Input: 2NFA $M = (Q, s, F, \delta)$
Output: DFA accepting $\mathcal{L}(M)$ having at most $2^{|Q|^2+|Q|}$ states.

How to collect all the information M can gather
about its input in a single sweep?

Tables



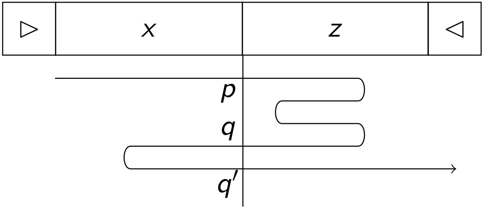
$$T : \Sigma^* \to 2^Q \times 2^{Q \times Q}$$

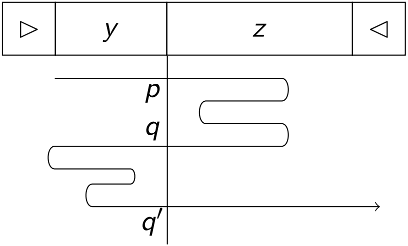
possible states when first entering z

enter/return relation when crossing from right

"$T\,x$ abstracts away the first part of the composite word xy."

Tables





finite type

$$T : \Sigma^* \to 2^Q \times 2^{Q \times Q}$$

possible states
when first entering z

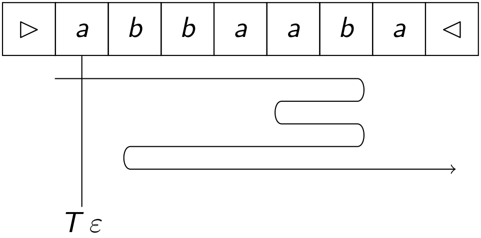
enter/return relation
when crossing from right

If $T\,x = T\,y$, then $xz \in \mathcal{L}(M)$ iff $yz \in \mathcal{L}(M)$

One-Way Sweep

SAARLAND UNIVERSITY
COMPUTER SCIENCE

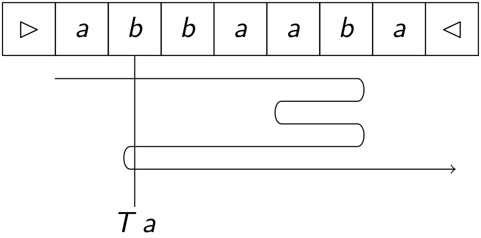
$$T : \Sigma^* \to 2^Q \times 2^{Q\times Q}$$



$$\begin{aligned} Q' &:= 2^Q \times 2^{Q\times Q} \\ s' &:= T\varepsilon \\ \delta'\,(Tx)\,a &:= T(xa) \\ F' &:= \{\, Tx \mid x \in \mathcal{L}(M) \,\} \end{aligned}$$

One-Way Sweep

$$T : \Sigma^* \to 2^Q \times 2^{Q \times Q}$$

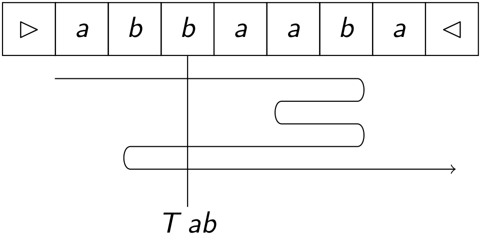


$$\begin{aligned} Q' &:= 2^Q \times 2^{Q \times Q} \\ s' &:= T\varepsilon \\ \delta'\,(Tx)\,a &:= T(xa) \\ F' &:= \{\, Tx \mid x \in \mathcal{L}(M) \,\} \end{aligned}$$

One-Way Sweep

SAARLAND UNIVERSITY
COMPUTER SCIENCE

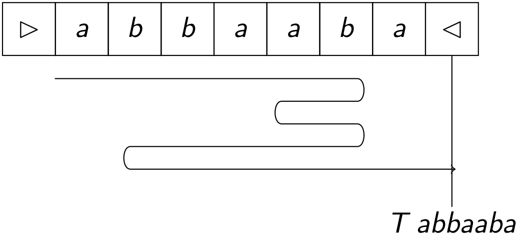
$$T : \Sigma^* \to 2^Q \times 2^{Q \times Q}$$



$$
\begin{aligned}
Q' &:= 2^Q \times 2^{Q \times Q} \\
s' &:= T\varepsilon \\
\delta'\,(Tx)\,a &:= T(xa) \\
F' &:= \{\, Tx \mid x \in \mathcal{L}(M) \,\}
\end{aligned}
$$

One-Way Sweep

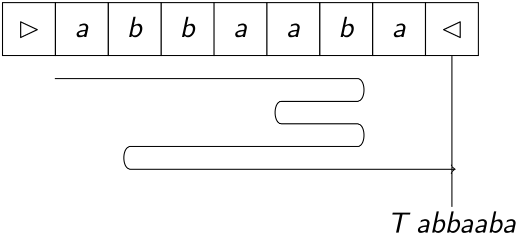
$$T : \Sigma^* \to 2^Q \times 2^{Q \times Q}$$



$$\begin{aligned} Q' &:= 2^Q \times 2^{Q \times Q} \\ s' &:= T\varepsilon \\ \delta'\,(Tx)\,a &:= T(xa) \\ F' &:= \{\, Tx \mid x \in \mathcal{L}(M) \,\} \end{aligned}$$

One-Way Sweep

$$T : \Sigma^* \to 2^Q \times 2^{Q \times Q}$$



$$\begin{aligned} Q' &:= 2^Q \times 2^{Q \times Q} \\ s' &:= T\varepsilon \\ \delta'\,(Tx)\,a &:= T(xa) \quad \text{surjectivity? right congruence?} \\ F' &:= \{\, Tx \mid x \in \mathcal{L}(M) \,\} \end{aligned}$$

Reduction to Myhill-Nerode

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Theorem (Myhill-Nerode)

Let L be a decidable language, X a finite type, and $T : \Sigma^* \to X$ such that

1. $T\,x = T\,y \to T(xa) = T(ya)$ *(T right congruent)*
2. $T\,x = T\,y \to (x \in L \leftrightarrow y \in L)$ *(T refines L)*

Then one can construct a DFA accepting L that has at most $|X|$ states.

Proof: construct DFA $A = (Q', s', F', \delta')$ where

$$\begin{aligned} Q' &:= X \\ s' &:= T\varepsilon \\ \delta'\,(Tx)\,a &:= T(xa) \\ F' &:= \{\, Tx \mid x \in L \,\} \end{aligned}$$

Reduction to Myhill-Nerode

SAARLAND UNIVERSITY COMPUTER SCIENCE

Theorem (Myhill-Nerode)

Let L be a decidable language, X a finite type, and $T : \Sigma^* \to X$ such that

1. $T\,x = T\,y \to T(xa) = T(ya)$ *(T right congruent)*
2. $T\,x = T\,y \to (x \in L \leftrightarrow y \in L)$ *(T refines L)*

Then one can construct a DFA accepting L that has at most $|X|$ states.

Proof: construct DFA $A = (Q', s', F', \delta')$ where

$$\begin{aligned} Q' &:= T(\Sigma^*) \\ s' &:= T\varepsilon \\ \delta'\,(Tx)\,a &:= T(xa) \\ F' &:= \{\, Tx \mid x \in L \,\} \end{aligned}$$

Reduction to Myhill-Nerode

Theorem (Myhill-Nerode)

Let L be a decidable language, X a finite type, and $T : \Sigma^* \to X$ such that

1. $T\,x = T\,y \to T(xa) = T(ya)$ *(T right congruent)*
2. $T\,x = T\,y \to (x \in L \leftrightarrow y \in L)$ *(T refines L)*

Then one can construct a DFA accepting L that has at most $|X|$ states.

Proof: construct DFA $A = (Q', s', F', \delta')$ where

$$\begin{aligned} Q' &:= T(\Sigma^*) \\ s' &:= T\varepsilon \\ \delta'\,q\,a &:= T((\overline{T}q)a)) \\ F' &:= \{\, Tx \mid x \in L \,\} \end{aligned}$$

where $\overline{T} : Q' \to \Sigma^*$ satisfies $T(\overline{T}q) = q$

Reduction to Myhill-Nerode

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Theorem (Myhill-Nerode)

Let L be a decidable language, X a finite type, and $T : \Sigma^* \to X$ such that

1. $T\,x = T\,y \to T(xa) = T(ya)$ *(T right congruent)*
2. $T\,x = T\,y \to (x \in L \leftrightarrow y \in L)$ *(T refines L)*

Then one can construct a DFA accepting L that has at most $|X|$ states.

Proof: construct DFA $A = (Q', s', F', \delta')$ where

$$\begin{aligned} Q' &:= T(\Sigma^*) \\ s' &:= T\varepsilon \\ \delta'\,q\,a &:= T((\overline{T}q)a)) \\ F' &:= \{\, q \mid \overline{T}\,q \in L \,\} \end{aligned}$$

where $\overline{T} : Q' \to \Sigma^*$ satisfies $T(\overline{T}q) = q$

Size Bounds

$$T : \Sigma^* \to 2^Q \times 2^{Q \times Q}$$

Lemma

T is right congruent and refines $\mathcal{L}(M)$.

Theorem (Vardi '89)

For every n-state 2NFA M one can construct a DFA accepting $\mathcal{L}(M)$ that has at most 2^{n^2+n} states.

If M is deterministic: $T \rightsquigarrow T' : \Sigma^* \to (Q+1) \times (Q \Rightarrow_{fin} Q+1)$

Theorem (Shepherdson '59)

For every n-state 2DFA M one can construct a DFA accepting $\mathcal{L}(M)$ that has at most $(n+1)^{(n+1)}$ states.

A Glimpse of Technicalities

SAARLAND UNIVERSITY COMPUTER SCIENCE

$$(p,i) \xrightarrow[x]{k} (q,j) \;:=\; (p,i) \xrightarrow[x]{} (q,j) \wedge i \neq k$$

$$Tx \;:=\; (\{\, q \mid (s,1) \xrightarrow[x]{|x|+1}{}^{*} (q,|x|+1) \,\}, \ldots)$$

Lemma

If $Tx = Ty$ then every run on xz that starts end ends on z has a corresponding run on yz.

Lemma

T is right congruent and refines $\mathcal{L}(M)$.

A Glimpse of Technicalities

$$(p,i) \xrightarrow[x]{k} (q,j) \;:=\; (p,i) \xrightarrow[x]{} (q,j) \wedge i \neq k$$

$$Tx \;:=\; (\{\, q \mid (s,1) \xrightarrow[x]{|x|+1}{}^{*} (q,|x|+1)\,\}, \ldots)$$

Lemma

If $Tx = Ty$, $i \leq |z| + 1$, $1 \leq j \leq |z| + 1$, and $1 \leq k$, then $(p, |x| + i) \xrightarrow[xz]{|x|+k}{}^{*} (q, |x| + j)$ iff $(p, |y| + i) \xrightarrow[xz]{|y|+k}{}^{*} (q, |y| + j)$.

Lemma

T is right congruent and refines $\mathcal{L}(M)$.

A Glimpse of Technicalities

SAARLAND UNIVERSITY
COMPUTER SCIENCE

$$(p,i) \xrightarrow[x]{k} (q,j) \ := (p,i) \xrightarrow[x]{} (q,j) \land i \neq k$$

$$Tx \ := (\{\, q \mid (s,1) \xrightarrow[x]{|x|+1}{}^{*} (q, |x|+1) \,\}, \ldots)$$

Lemma

If $Tx = Ty$, $i \leq |z| + 1$, $1 \leq j \leq |z| + 1$, *and* $1 \leq k$, *then* $(p, |x| + i) \xrightarrow[xz]{|x|+k}{}^{*} (q, |x| + j)$ *iff* $(p, |y| + i) \xrightarrow[xz]{|y|+k}{}^{*} (q, |y| + j)$.

Lemma

T *is right congruent and refines* $\mathcal{L}(M)$.

Taming Dependent Types

$$C_x := Q \times \{\, i : \mathbb{N} \mid i < |x| + 2 \,\}$$

$$\_ \underset{\_}{\rightarrow} \_ : \forall x.\, C_x \to C_x \to \mathbb{B}$$

$$(p, i) \underset{x}{\rightarrow} (q, i + 1)$$

$$(p, \langle i, H_1 \rangle) \underset{x}{\rightarrow} (q, \langle i + 1, H_2 \rangle) \qquad H_1 : i < |x| + 2, H_2 : i + 1 < |x| + 2$$

Taming Dependent Types

SAARLAND UNIVERSITY
COMPUTER SCIENCE

$$C_x := Q \times \{\, i : \mathbb{N} \mid i < |x| + 2 \,\}$$
$$\_ \underset{\_}{\rightarrow} \_ : \forall x.\, C_x \rightarrow C_x \rightarrow \mathbb{B}$$

$$(p, i) \underset{x}{\rightarrow} (q, i + 1)$$

$$(p, \langle i, H_1 \rangle) \underset{x}{\rightarrow} (q, \langle i + 1, H_2 \rangle) \qquad H_1 : i < |x| + 2, H_2 : i + 1 < |x| + 2$$

$$(p, \mathrm{inord}\, i) \underset{x}{\rightarrow} (q, \mathrm{inord}\,(i + 1)) \qquad \mathrm{inord} : \mathbb{N} \rightarrow \{\, i : \mathbb{N} \mid i < |x| + 2 \,\}$$

Maintains the separation between stating properties and proving the bounds

Formalization

Formalization:

Part	**LoC**
2FAs and simulation of DFAs	160
Vardi construction (incl. runs)	150
Shepherdson construction (2NFAs and 2DFAs)	290
Total	600

Prerequisites:

- Coq: dependent types, types as first class objects
- Ssreflect: discrete types, finite types, finite sets, quotients, ...
- Theory: DFAs, NFAs, (Myhill-Nerode) (Doczkal Kaiser Smolka '13)

Formalization

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Formalization:

Part	LoC
1FAs and Myhill-Nerode	600
2FAs and simulation of DFAs	160
Vardi construction (incl. runs)	150
Shepherdson construction (2NFAs and 2DFAs)	290
Total	1200

Prerequisites:

Coq dependent types, types as first class objects

Ssreflect discrete types, finite types, finite sets, quotients, ...

Theory DFAs, NFAs, (Myhill-Nerode) (Doczkal Kaiser Smolka '13)

Conclusion & Future Work

Conclusion:

- Translations from 2FAs to 1FAs defined and verified constructively
- Translation employs constructive variant of Myhill-Nerode
- Correctness proofs are relatively short but technical

`https://www.ps.uni-saarland.de/extras/itp16-2FA`

Future Work:

- Translation from 2NFAs to NFAs (Kapoutsis '05)
- other automata models: infinite words, infinite trees, alternating, ...

Conclusion & Future Work

Conclusion:

- Translations from 2FAs to 1FAs defined and verified constructively
- Translation employs constructive variant of Myhill-Nerode
- Correctness proofs are relatively short but technical

`https://www.ps.uni-saarland.de/extras/itp16-2FA`

Future Work:

- Translation from 2NFAs to NFAs (Kapoutsis '05)
- other automata models: infinite words, infinite trees, alternating, ...

Thank You! Questions?