The Expressive Power of Linear-time Temporal Logic

K Narayan Kumar

Chennai Mathematical Institute
email: kumar@cmi.ac.in

IIT Guwahati, July 2006
LTL is expressible in FO.
Summary of Last Lecture

- LTL is expressible in FO.
- FO definable languages are regular. (Via EF Games)
Summary of Last Lecture

- LTL is expressible in FO.
- FO definable languages are regular. (Via EF Games)
- FO definable languages are aperiodic. (Via EF Games, Syntactic Monoid)
Star-free Regular Languages

Regular expressions constructed without the * operator:

\[ e ::= a \mid e_1 + e_2 \mid \neg e_1 \mid e_1 . e_2 \]
Star-free Regular Languages

Regular expressions constructed without the $\ast$ operator:

$$e ::= a \mid e_1 + e_2 \mid \neg e_1 \mid e_1.e_2$$

**Theorem:** (Schutzenberger) $L$ is aperiodic if and only if it is star-free.

**Theorem:** (McNaughton and Papert) $L$ is star-free if and only if it is FO expressible.
Star-free Regular Languages

Regular expressions constructed without the $*$ operator:

$$e ::= a \mid e_1 + e_2 \mid \neg e_1 \mid e_1 . e_2$$

**Theorem:** (Schutzenberger) $L$ is aperiodic if and only if it is star-free.

**Theorem:** (McNaughton and Papert) $L$ is star-free if and only if it is FO expressible.

**Question:** Can we translate star-free expressions into LTL?
Star-free Regular Languages

Regular expressions constructed without the $\ast$ operator:

$$e ::= a \mid e_1 + e_2 \mid \neg e_1 \mid e_1.e_2$$

**Theorem:** (Schützenberger) $L$ is aperiodic if and only if it is star-free.

**Theorem:** (McNaughton and Papert) $L$ is star-free if and only if it is FO expressible.

**Question:** Can we translate star-free expressions into LTL?

How do we put together LTL formulas $\varphi_1$ and $\varphi_2$ to describe the language $L(\varphi_1).L(\varphi_2)$?
Star-free Regular Languages

Regular expressions constructed without the $\ast$ operator:

$$ e ::= a \mid e_1 + e_2 \mid \neg e_1 \mid e_1.e_2 $$

**Theorem:** (Schützenberger) $L$ is aperiodic if and only if it is star-free.

**Theorem:** (McNaughton and Papert) $L$ is star-free if and only if it is FO expressible.

**Question:** Can we translate star-free expressions into LTL?

How do we put together LTL formulas $\varphi_1$ and $\varphi_2$ to describe the language $L(\varphi_1).L(\varphi_2)$?

Easy if the decomposition is unambiguous. (eg.) $L_1.c.L_2$ where either $L_1$ or $L_2$ is c-free.
The proof proceeds via a double induction: On the size of the monoid recognizing $L$ and the size of the alphabet.
The Proof: Base cases

The proof proceeds via a double induction: On the size of the monoid recognizing \( L \) and the size of the alphabet.

The Base Cases:
The proof proceeds via a double induction: On the size of the monoid recognizing $L$ and the size of the alphabet.

The Base Cases:
- $M$ is the trivial monoid.
The Proof: Base cases

The proof proceeds via a double induction: On the size of the monoid recognizing $L$ and the size of the alphabet.

The Base Cases:

- $M$ is the trivial monoid.
  - $L$ is $\Sigma^+$. Use $\top$. 
The Proof: Base cases

The proof proceeds via a double induction: On the size of the monoid recognizing $L$ and the size of the alphabet.

The Base Cases:

- $M$ is the trivial monoid.
  - $L$ is $\Sigma^+$. Use $\top$.
  - $L$ is $\emptyset$. Use $\bot$. 
The Proof: Base cases

The proof proceeds via a double induction: On the size of the monoid recognizing $L$ and the size of the alphabet.

The Base Cases:

- $M$ is the trivial monoid.
  - $L$ is $\Sigma^+$. Use $\top$.
  - $L$ is $\emptyset$. Use $\bot$.
- $\Sigma$ is singleton.
The Proof: Base cases

The proof proceeds via a double induction: On the size of the monoid recognizing $L$ and the size of the alphabet.

The Base Cases:
- $M$ is the trivial monoid.
  - $L$ is $\Sigma^+$. Use $\top$.
  - $L$ is $\emptyset$. Use $\bot$.
- $\Sigma$ is singleton.
  - $L$ is finite. Easy.
The Proof: Base cases

The proof proceeds via a double induction: On the size of the monoid recognizing $L$ and the size of the alphabet.

The Base Cases:

- **$M$** is the trivial monoid.
  - $L$ is $\Sigma^+$. Use $\top$.
  - $L$ is $\emptyset$. Use $\bot$.
- **$\Sigma$** is singleton.
  - $L$ is finite. Easy.
  - $L$ is $\{a^i \mid i \geq N\}$. Easy.
The Proof:

**Induction Step:** Given $L$ over an alphabet $\Sigma$ recognized by a monoid $M$ such that:
The Proof:

Induction Step: Given $L$ over an alphabet $\Sigma$ recognized by a monoid $M$ such that:

- if $|M'| < |M|$ then any language recognized by $M'$ is expressible in $LTL$. 

K Narayan Kumar

The Expressive Power of Linear-time Temporal Logic
The Proof:

**Induction Step:** Given $L$ over an alphabet $\Sigma$ recognized by a monoid $M$ such that:

- if $|M'| < |M|$ then any language recognized by $M'$ is expressible in $LTL$.
- if $L'$ is a language over an alphabet $A$ with $|A| < |\Sigma|$ recognized by $M$ then $L'$ is expressible in $LTL_A$.

show that $L$ is expressible in $LTL_\Sigma$. 
The Proof:

Induction Step: Given $L$ over an alphabet $\Sigma$ recognized by a monoid $M$ such that:

- if $|M'| < |M|$ then any language recognized by $M'$ is expressible in $LTL$.
- if $L'$ is a language over an alphabet $A$ with $|A| < |\Sigma|$ recognized by $M$ then $L'$ is expressible in $LTL_A$.

show that $L$ is expressible in $LTL_{\Sigma}$.

Observation 1: If $\varphi$ is a $LTL_A$ formula describing the language $L$ and $A \subseteq \Sigma$ then

$$\varphi \land \bigwedge_{a \in \Sigma \setminus A} G \neg a$$

is a $LTL_{\Sigma}$ formula that describes $L$. 
Let $L$ be recognized by $M$ via the morphism $h$ as $h^{-1}(X)$. 
Let $L$ be recognized by $M$ via the morphism $h$ as $h^{-1}(X)$.

Pick a letter $c$ such that $h(c) \neq 1$. 
Splitting by a letter

Let \( L \) be recognized by \( M \) via the morphism \( h \) as \( h^{-1}(X) \).

Pick a letter \( c \) such that \( h(c) \neq 1 \).

Such a \( c \) must exist. Otherwise, \( L \) is recognized by the trivial monoid.
Splitting by a letter

Let $L$ be recognized by $M$ via the morphism $h$ as $h^{-1}(X)$.

Pick a letter $c$ such that $h(c) \neq 1$.

Such a $c$ must exist. Otherwise, $L$ is recognized by the trivial monoid.

Decompose $L$ into three disjoint sets:

- $L_0$ consisting of words of $L$ with no $c$s.
- $L_1$ consisting of words of $L$ with exactly one $c$.
- $L_2$ consisting of words of $L$ with at least two $c$s.
Splitting by a letter

Let $L$ be recognized by $M$ via the morphism $h$ as $h^{-1}(X)$.

Pick a letter $c$ such that $h(c) \neq 1$.

_Such a $c$ must exist. Otherwise, $L$ is recognized by the trivial monoid._

Decompose $L$ into three disjoint sets:

- $L_0$ consisting of words of $L$ with no $c$s.
- $L_1$ consisting of words of $L$ with exactly one $c$.
- $L_2$ consisting of words of $L$ with at least two $c$s.

“No cs”, “Exactly 1 c” and “Atleast 2 cs” are expressible in LTL.
Splitting by a letter

Let $L$ be recognized by $M$ via the morphism $h$ as $h^{-1}(X)$.

Pick a letter $c$ such that $h(c) \neq 1$.

Such a $c$ must exist. Otherwise, $L$ is recognized by the trivial monoid.

Decompose $L$ into three disjoint sets:

- $L_0$ consisting of words of $L$ with no $c$s.
- $L_1$ consisting of words of $L$ with exactly one $c$.
- $L_2$ consisting of words of $L$ with at least two $c$s.

“No cs”, “Exactly 1 c” and “Atleast 2 cs” are expressible in LTL.

It suffices to show that each of these three languages is LTL expressible.
Let $A = \Sigma \setminus \{c\}$. 
Let $A = \Sigma \setminus \{c\}$.

- $L_0$ is language over a smaller alphabet $A$, recognized by $M$ via $h$. 

Let $A = \Sigma \setminus \{c\}$.

- $L_0$ is language over a smaller alphabet $A$, recognized by $M$ via \( h \).
- So, $L_0$ is defined by an $LTL_A$ formula $\varphi_0$ over $A$. 

The Trivial Case: $L_0$
The Trivial Case: $L_0$

Let $A = \Sigma \setminus \{c\}$.

- $L_0$ is language over a smaller alphabet $A$, recognized by $M$ via $h$.
- So, $L_0$ is defined by an $LTL_A$ formula $\phi_0$ over $A$.
- By Observation 1, it is expressible in $LTL_{\Sigma}$.
The Easy Case: $L_1$

\[
L_1 = \bigcup \alpha. h(c). \beta \in X (h^{-1}(\alpha) \cap A^*).c.(h^{-1}(\beta) \cap A^*)
\]
The Easy Case: $L_1$

\[ L_1 = \bigcup_{\alpha \cdot h(c) \cdot \beta \in X} (h^{-1}(\alpha) \cap A^*) \cdot c \cdot (h^{-1}(\beta) \cap A^*) \]

Why?
The Easy Case: $L_1$

\[ L_1 = \bigcup_{\alpha. \quad h(c). \quad \beta \in X} (h^{-1}(\alpha) \cap A^*).c.(h^{-1}(\beta) \cap A^*) \]

Why?

- If $xcy$ is in the RHS then $h(xcy) = \alpha. h(c). \beta \in X$. Thus $xcy \in L$. 

The Easy Case: $L_1$

$$L_1 = \bigcup_{\alpha. h(c). \beta \in X} \left( h^{-1}(\alpha) \cap A^* \right). c. \left( h^{-1}(\beta) \cap A^* \right)$$

Why?

- If $xcy$ is in the RHS then $h(xcy) = \alpha. h(c). \beta \in X$. Thus $xcy \in L$.
- Let $w \in L_1$. Therefore, $w = xcy$. Take $\alpha = h(x)$ and $\beta = h(y)$. 
The Easy Case: $L_1$

\[ L_1 = \bigcup_{\alpha, h(c), \beta \in X} (h^{-1}(\alpha) \cap A^*).c.(h^{-1}(\beta) \cap A^*) \]

Let $L_\alpha = h^{-1}(\alpha) \cap A^*$ and $L_\beta = h^{-1}(\beta) \cap A^*$. 
The Easy Case: \( L_1 \)

\[
L_1 = \bigcup_{\alpha, h(c), \beta \in X} (h^{-1}(\alpha) \cap A^*).c.(h^{-1}(\beta) \cap A^*)
\]

Let \( L_\alpha = h^{-1}(\alpha) \cap A^* \) and \( L_\beta = h^{-1}(\beta) \cap A^* \).

\( L_1 \) is a union of languages of the form \( L_\alpha.c.L_\beta \) where \( L_\alpha, L_\beta \subseteq A^* \) are recognized by \( M \) and hence \( LTL_A \) (and therefore \( LTL_\Sigma \)) expressible.
The Easy Case: $L_1$

$$L_1 = \bigcup_{\alpha, h(c), \beta \in X} (h^{-1}(\alpha) \cap A^*).c.(h^{-1}(\beta) \cap A^*)$$

Let $L_\alpha = h^{-1}(\alpha) \cap A^*$ and $L_\beta = h^{-1}(\beta) \cap A^*$.

$L_1$ is a union of languages of the form $L_\alpha . c . L_\beta$ where $L_\alpha, L_\beta \subseteq A^*$ are recognized by $M$ and hence $\text{LTL}_A$ (and therefore $\text{LTL}_\Sigma$) expressible.

Well, almost! $L_\alpha \cap A^+$ and $L_\beta \cap A^+$ are LTL expressible. We have to deal with $\epsilon$ separately.
We may rewrite $L_\alpha \cdot c \cdot L_\beta$ as

$$A^* \cdot c \cdot L_\beta \cap L_\alpha \cdot c \cdot \Sigma^*$$
Dealing with Unambiguous Concatenations

We may rewrite $L_\alpha.c.L_\beta$ as

$$A^*.c.L_\beta \cap L_\alpha.c.\Sigma^*$$

If $\varphi_\beta$ is the $\text{LTL}_\Sigma$ formula expressing $L_\beta \cap A^+$ then $\varphi_1 = \top U (c \land X \varphi_\beta)$ describes $A^*.c.(L_\beta \cap A^+)$. 
We may rewrite $L_\alpha . c . L_\beta$ as

$$A^* . c . L_\beta \cap L_\alpha . c . \Sigma^*$$

If $\varphi_\beta$ is the $LTL_\Sigma$ formula expressing $L_\beta \cap A^+$ then

$$\varphi_1 = \top U (c \land X \varphi_\beta)$$

describes $A^* . c . (L_\beta \cap A^+)$. If $\epsilon \notin L_\beta$ then $\varphi_1$ also describes the language $A^* . c . L_\beta$. 
Dealing with Unambiguous Concatenations

We may rewrite $L_\alpha c L_\beta$ as

$$A^* c L_\beta \cap L_\alpha c \Sigma^*$$

If $\varphi_\beta$ is the $LTL_\Sigma$ formula expressing $L_\beta \cap A^+$ then

$$\varphi_1 = \top U (c \land X \varphi_\beta)$$
describes $A^* c (L_\beta \cap A^+)$. If $\epsilon \notin L_\beta$ then $\varphi_1$ also describes the language $A^* c L_\beta$.

Otherwise, $\varphi_1 \lor \top U (c \land \neg X \top)$ describes the language $A^* c L_\beta$. 
We may rewrite $L_\alpha . c . L_\beta$ as

$$A^* . c . L_\beta \cap L_\alpha . c . \Sigma^*$$

If $\varphi_\beta$ is the $LTL_\Sigma$ formula expressing $L_\beta \cap A^+$ then

$\varphi_1 = \top U (c \land X \varphi_\beta)$ describes $A^* . c . (L_\beta \cap A^+)$.

If $\epsilon \notin L_\beta$ then $\varphi_1$ also describes the language $A^* . c . L_\beta$.

Otherwise, $\varphi_1 \lor \top U (c \land \neg X \top)$ describes the language $A^* . c . L_\beta$.

This case was easy because our modalities walk only to the right and so cannot “stray” to the left. Dealing with $L_\alpha . c . \Sigma^*$ will need a little more work.
Unambiguous Concatenation: $L_\alpha . c . \Sigma^*$

Let $\varphi_\alpha$ be a $LTL_A$ formula describing $L_\alpha \cap A^+$. 
Let $\varphi_\alpha$ be a $LTL_A$ formula describing $L_\alpha \cap A^+$. We cannot use $\varphi_\alpha$ to describe $L_\alpha . c . \Sigma^*$ since the modalities may walk to the right and cross the $c$ boundary.
Let $\varphi_\alpha$ be a $LTL_A$ formula describing $L_\alpha \cap A^+$. We “relativize” $\varphi_\alpha$ to a formula $\varphi'_\alpha$ which examines the part to the left of the first $c$ and checks if it satisfies $\varphi_\alpha$. 
Let $\varphi_\alpha$ be a $LTL_A$ formula describing $L_\alpha \cap A^+$. We “relativize” $\varphi_\alpha$ to a formula $\varphi'_\alpha$ which examines the part to the left of the first $c$ and checks if it satisfies $\varphi_\alpha$. Formally, $w \models \varphi'_\alpha$ iff $w = xcy$, $x \in A^+$ and $x \models \varphi_\alpha$. 
Let $\varphi_\alpha$ be a $LTL_A$ formula describing $L_{\alpha} \cap A^+$. 

We “relativize” $\varphi_\alpha$ to a formula $\varphi'_\alpha$ which examines the part to the left of the first $c$ and checks if it satisfies $\varphi_\alpha$. 

Formally, $w \models \varphi'_\alpha$ iff $w = xcy$, $x \in A^+$ and $x \models \varphi_\alpha$. 

This relativization is defined via structural recursion as follows:

$$
\begin{align*}
  a' &= a \land XFc \\
  (\varphi \land \psi)' &= \varphi' \land \psi' \\
  (\neg \varphi)' &= (\neg \varphi') \land \neg c \land Fc \\
  (\varphi XU \psi)' &= (\varphi' \land \neg c) XU (\psi' \land \neg c)
\end{align*}
$$
Unambiguous Concatenation: $L_\alpha.c.\Sigma^*$

Let $\varphi_\alpha$ be a $LTL_A$ formula describing $L_\alpha \cap A^+$. 

We “relativize” $\varphi_\alpha$ to a formula $\varphi'_\alpha$ which examines the part to the left of the first $c$ and checks if it satisfies $\varphi_\alpha$.

Formally, $w \models \varphi'_\alpha$ iff $w = xcy$, $x \in A^+$ and $x \models \varphi_\alpha$.

This relativization is defined via structural recursion as follows:

\[
\begin{align*}
a' &= a \land XFc \\
(\varphi \land \psi)' &= \varphi' \land \psi' \\
(\neg \varphi)' &= (\neg \varphi') \land \neg c \land Fc \\
(\varphi XU \psi)' &= (\varphi' \land \neg c) XU (\psi' \land \neg c)
\end{align*}
\]

$\varphi_2 = \varphi'_\alpha$ describes $(L_\alpha \cap A^+).c.\Sigma^*$. If $\epsilon \not\in L_\alpha$ then $\varphi_2$ also describes $L_\alpha.c.\Sigma^*$. Otherwise, use $\varphi_2 \lor c$. 

K Narayan Kumar  The Expressive Power of Linear-time Temporal Logic
I WILL BE SLOPPY WITH $\epsilon$
FROM NOW ON.
The Interesting Case: $L_2$

So far, we got away by examining the alphabet. Here we need to examine $M$ and induct on its size.
The Interesting Case: \( L_2 \)

So far, we got away by examining the alphabet. Here we need to examine \( M \) and induct on its size.

A word \( w \) in \( L_2 \) is of the form \( t_0ct_1ct_2c \ldots t_{k-1}ct_k \) for some \( k > 1 \), \( t_i \in A^* \).
The Interesting Case: $L_2$

So far, we got away by examining the alphabet. Here we need to examine $M$ and induct on its size.

A word $w$ in $L_2$ is of the form $t_0ct_1ct_2c \ldots t_{k-1}ct_k$ for some $k > 1$, $t_i \in A^*$.

Further, $h(w) = h(t_0)h(ct_1ct_2ct_3 \ldots t_{k-1}c)h(t_k) \in X$. 
The Interesting Case: $L_2$

So far, we got away by examining the alphabet. Here we need to examine $M$ and induct on its size.

A word $w$ in $L_2$ is of the form $t_0ct_1ct_2c\ldots t_{k-1}ct_k$ for some $k > 1$, $t_i \in A^*$.

Further, $h(w) = h(t_0)h(ct_1ct_2ct_3\ldots t_{k-1}c)h(t_k) \in X$.

Let $\Delta = (cA^*)^+c$. Then, $L_2 \subseteq A^*.\Delta.A^*$. 
The Interesting Case: \( L_2 \)

So far, we got away by examining the alphabet. Here we need to examine \( M \) and induct on its size.

A word \( w \) in \( L_2 \) is of the form \( t_0ct_1ct_2c\ldots t_{k-1}ct_k \) for some \( k > 1 \), \( t_i \in A^* \).

Further, \( h(w) = h(t_0)h(ct_1ct_2ct_3\ldots t_{k-1}c)h(t_k) \in X \).

Let \( \Delta = (cA^*)^+c \). Then, \( L_2 \subseteq A^*.\Delta.A^* \).

\[
L_2 = \bigcup_{\alpha,\beta,\gamma \in X} (h^{-1}(\alpha) \cap A^*).(h^{-1}(\beta) \cap \Delta).(h^{-1}(\gamma) \cap A^*)
\]
The Interesting Case: $L_2$

So far, we got away by examining the alphabet. Here we need to examine $M$ and induct on its size.

A word $w$ in $L_2$ is of the form $t_0ct_1ct_2c\ldots t_{k-1}ct_k$ for some $k > 1$, $t_i \in A^*$.

Further, $h(w) = h(t_0)h(ct_1ct_2ct_3\ldots t_{k-1}c)h(t_k) \in X$.

Let $\Delta = (cA^*)^+c$. Then, $L_2 \subseteq A^*.\Delta.A^*$.

$$L_2 = \bigcup_{\alpha\beta\gamma \in X} (h^{-1}(\alpha) \cap A^*).(h^{-1}(\beta) \cap \Delta).(h^{-1}(\gamma) \cap A^*)$$

The first and third components are LTL definable. What about the middle component?
An Outline of the proof

We show that the language $L_{\beta} \cap \Delta$ is LTL definable as follows:
An Outline of the proof

We show that the language $L_\beta \cap \Delta$ is LTL definable as follows:

1. Translate each word in $\Delta$ to a word over the alphabet $M$ (actually $h(A^*) \subseteq M$) via a map $\sigma$. 
An Outline of the proof

We show that the language $L_\beta \cap \Delta$ is LTL definable as follows:

1. Translate each word in $\Delta$ to a word over the alphabet $M$ (actually $h(A^*) \subseteq M$) via a map $\sigma$.
2. Construct a language $K$ over $M$ such that:
An Outline of the proof

We show that the language $L_\beta \cap \Delta$ is LTL definable as follows:

1. Translate each word in $\Delta$ to a word over the alphabet $M$ (actually $h(A^*) \subseteq M$) via a map $\sigma$.

2. Construct a language $K$ over $M$ such that:
   
   $\sigma^{-1}(K) = L_\beta \cap \Delta$
An Outline of the proof

We show that the language $L_\beta \cap \Delta$ is LTL definable as follows:

1. Translate each word in $\Delta$ to a word over the alphabet $M$ (actually $h(A^*) \subseteq M$) via a map $\sigma$.
2. Construct a language $K$ over $M$ such that:
   1. $\sigma^{-1}(K) = L_\beta \cap \Delta$
   2. $K$ is recognized by a aperiodic monoid smaller than $M$. 
An Outline of the proof

We show that the language $L_\beta \cap \Delta$ is LTL definable as follows:

1. Translate each word in $\Delta$ to a word over the alphabet $M$ (actually $h(A^*) \subseteq M$) via a map $\sigma$.
2. Construct a language $K$ over $M$ such that:
   1. $\sigma^{-1}(K) = L_\beta \cap \Delta$
   2. $K$ is recognized by a aperiodic monoid smaller than $M$.
   3. the $LTL_M$ formula describing $K$ can be lifted to a formula in $LTL_\Sigma$ describing $L_\beta \cap \Delta$. 
An Outline of the proof

We show that the language $L_\beta \cap \Delta$ is LTL definable as follows:

1. Translate each word in $\Delta$ to a word over the alphabet $M$ (actually $h(A^*) \subseteq M$) via a map $\sigma$.
2. Construct a language $K$ over $M$ such that:
   1. $\sigma^{-1}(K) = L_\beta \cap \Delta$
   2. $K$ is recognized by a aperiodic monoid smaller than $M$.
   3. the $LTL_M$ formula describing $K$ can be lifted to a formula in $LTL_\Sigma$ describing $L_\beta \cap \Delta$.

We use $m$ to denote elements of $M$ when treated as letters and $m$ when they are treated as elements of the monoid $M$. 
The map $\sigma$ and Language $K$

The map $\sigma$ is the obvious one:

$$\sigma c t_1 c t_2 \ldots t_{k-2} c t_{k-1} c = h(t_1) h(t_2) \ldots h(t_{k-1})$$
The map $\sigma$ and Language $K$

The map $\sigma$ is the obvious one:

$$\sigma ct_1 ct_2 \ldots t_{k-2} ct_{k-1} c = h(t_1) h(t_2) \ldots h(t_{k-1})$$

Given the map $\sigma$ and requirement 2.1, the definition of $K$ is also quite obvious:

$$K = \{m_1 m_2 \ldots m_k \mid h(c)m_1 h(c)m_2 \ldots h(c)m_k h(c) = \beta\}$$
The map $\sigma$ and Language $K$

The map $\sigma$ is the obvious one:

$$\sigma ct_1 ct_2 \ldots t_{k-2} ct_{k-1} c = h(t_1) h(t_2) \ldots h(t_{k-1})$$

Given the map $\sigma$ and requirement 2.1, the definition of $K$ is also quite obvious:

$$K = \{ m_1 m_2 \ldots m_k \mid h(c)m_1 h(c)m_2 \ldots h(c)m_k h(c) = \beta \}$$

With these definitions:

$$\sigma^{-1}(K) = \{ ct_1 ct_2 \ldots ct_k c \mid h(t_1) h(t_2) \ldots h(t_k) \in K \}$$
The map $\sigma$ and Language $K$

The map $\sigma$ is the obvious one:

$$\sigma ct_1 ct_2 \cdots t_{k-2} ct_{k-1} c = h(t_1) h(t_2) \cdots h(t_{k-1})$$

Given the map $\sigma$ and requirement 2.1, the definition of $K$ is also quite obvious:

$$K = \{ m_1 m_2 \cdots m_k \mid h(c) m_1 h(c) m_2 \cdots h(c) m_k h(c) = \beta \}$$

With these definitions:

$$\sigma^{-1}(K) = \{ ct_1 ct_2 \cdots ct_k c \mid h(t_1) h(t_2) \cdots h(t_k) \in K \}$$

$$= \{ ct_1 ct_2 \cdots ct_k c \mid h(c) h(t_1) h(c) h(t_2) \cdots h(c) h(t_k) h(c) = \beta \}$$
The map $\sigma$ and Language $K$

The map $\sigma$ is the obvious one:

$$\sigma ct_1 ct_2 \ldots t_{k-2} ct_{k-1} c = h(t_1)h(t_2)\ldots h(t_{k-1})$$

Given the map $\sigma$ and requirement 2.1, the definition of $K$ is also quite obvious:

$$K = \{m_1 m_2 \ldots m_k \mid h(c)m_1 h(c)m_2 \ldots h(c)m_k h(c) = \beta\}$$

With these definitions:

$$\sigma^{-1}(K) = \{ct_1 ct_2 \ldots ct_k c \mid h(t_1)h(t_2)\ldots h(t_k) \in K\}$$

$$= \{ct_1 ct_2 \ldots ct_k c \mid h(c)h(t_1)h(c)h(t_2)\ldots h(c)h(t_k)h(c) = \beta\}$$

$$= L_\beta \cap \Delta \text{ as required by 2.1}$$
Localizing a Monoid at an element

The following construction is due to Diekert and Gastin.

**The Monoid \( \text{Loc}_m(M) \):** Let \( M \) be a monoid and \( m \in M \). Then

\[
\text{Loc}_m(M) = (mM \cap Mm, \circ, m)
\]

where \((xm) \circ (ym) \triangleq xmy\).
Localizing a Monoid at an element

The following construction is due to Diekert and Gastin.

The Monoid $\text{Loc}_m(M)$: Let $M$ be a monoid and $m \in M$. Then

$$\text{Loc}_m(M) = (mM \cap Mm, \circ, m)$$

where $(xm) \circ (my) \triangleq xmy$.

- Observe that $xm \circ ym = xm \circ my' = xmy' = xym$. Thus $\circ$ is associative and $m = 1.m$ is the identity w.r.t. $\circ$. 

Localizing a Monoid at an element

The following construction is due to Diekert and Gastin.

The Monoid $\text{Loc}_m(M)$: Let $M$ be a monoid and $m \in M$. Then

$$\text{Loc}_m(M) = (mM \cap Mm, \circ, m)$$

where $(xm) \circ (my) \triangleq xmy$.

- Observe that $xm \circ ym = xm \circ my' = xmy' = xym$. Thus $\circ$ is associative and $m = 1.m$ is the identity w.r.t. $\circ$.
- $xm \circ xm \circ \ldots \circ xm = x^Nm$. Thus, $\text{Loc}_m(M)$ is aperiodic whenever $M$ is aperiodic.
Localizing a Monoid at an element

The following construction is due to Diekert and Gastin.

The Monoid $\text{Loc}_m(M)$: Let $M$ be a monoid and $m \in M$. Then

$$\text{Loc}_m(M) = (mM \cap Mm, \circ, m)$$

where $(xm) \circ (my) \triangleq xmy$.

- Observe that $xm \circ ym = xm \circ my' = xmy' = xym$. Thus $\circ$ is associative and $m = 1.m$ is the identity w.r.t. $\circ$.
- $xm \circ xm \circ \ldots \circ xm = x^N m$. Thus, $\text{Loc}_m(M)$ is aperiodic whenever $M$ is aperiodic.
- $1 \not\in \text{Loc}_m(M)$ if $m \neq 1$. This follows from the fact that $1 \neq m'm$ for any $m, m' \neq 1$. 
We now show that the monoid $\text{Loc}_{h(c)}(M)$ accepts the language $K$. 
We now show that the monoid $\text{Loc}_{h(c)}(M)$ accepts the language $K$.

Let $g : M^* \rightarrow \text{Loc}_{h(c)}(M)$ be given by $g(m) = h(c)mh(c)$. 
We now show that the monoid $\text{Loc}_{h(c)}(M)$ accepts the language $K$.

Let $g : M^* \rightarrow \text{Loc}_{h(c)}(M)$ be given by $g(m) = h(c)mh(c)$.

Claim: $K = g^{-1}(\beta)$
A Monoid for $K$

We now show that the monoid $\text{Loc}_{h(c)}(M)$ accepts the language $K$.

Let $g : M^* \rightarrow \text{Loc}_{h(c)}(M)$ be given by $g(m) = h(c)mh(c)$.

Claim: $K = g^{-1}(\beta)$

Proof:
A Monoid for $K$

We now show that the monoid $\text{Loc}_{h(c)}(M)$ accepts the language $K$.

Let $g : M^* \rightarrow \text{Loc}_{h(c)}(M)$ be given by $g(m) = h(c)m h(c)$.

Claim: $K = g^{-1}(\beta)$

Proof:

- Note that $\beta \in \text{Loc}_{h(c)}(M)$ whenever $h^{-1}(\beta) \cap \Delta \neq \emptyset$. 

A Monoid for $K$

We now show that the monoid $\text{Loc}_{h(c)}(M)$ accepts the language $K$.

Let $g : M^* \longrightarrow \text{Loc}_{h(c)}(M)$ be given by $g(m) = h(c)m h(c)$.

Claim: $K = g^{-1}(\beta)$

Proof:
- Note that $\beta \in \text{Loc}_{h(c)}(M)$ whenever $h^{-1}(\beta) \cap \Delta \neq \emptyset$.
- $g(m_1 m_2 \ldots m_k) = \beta$ if and only if $h(c)m_1 h(c) \circ h(c)m_2 h(c) \circ \ldots \circ h(c)m_k h(c) = \beta$ if and only if $h(c)m_1 h(c)m_2 h(c) \ldots h(c)m_k h(c) = \beta$ if and only if $m_1 m_2 \ldots m_k \in K$. 
We now show that the monoid $\text{Loc}_{h(c)}(M)$ accepts the language $K$.

Let $g : M^* \rightarrow \text{Loc}_{h(c)}(M)$ be given by $g(m) = h(c)mh(c)$.

Claim: $K = g^{-1}(\beta)$

Proof:

- Note that $\beta \in \text{Loc}_{h(c)}(M)$ whenever $h^{-1}(\beta) \cap \Delta \neq \emptyset$.
- $g(m_1m_2\ldots m_k) = \beta$ if and only if $h(c)m_1h(c) \circ h(c)m_2h(c) \circ \ldots \circ h(c)m_kh(c) = \beta$ if and only if $m_1m_2\ldots m_k \in K$.

$K$ is recognized by a smaller monoid and hence there is an $LTL_M$ formula that describes $K$. 
Lifting the formula for $K$

We show that for any formula $\varphi$ in $LTL_M$, there is a formula $\varphi^\#$ in $LTL_\Sigma$ such that

$$w \models \varphi^\# \iff w = ct_1 ct_2 \ldots t_{k-1} ct_k, \text{ with } t_i \in A^* \text{ and } \sigma(ct_1 ct_2 \ldots t_{k-1} c) \models \varphi$$
Lifting the formula for $K$

We show that for any formula $\varphi$ in $LTL_M$, there is a formula $\varphi^\#$ in $LTL_\Sigma$ such that

$$w \models \varphi^\# \iff w = ct_1ct_2c\ldots t_{k-1}ct_k, \text{ with } t_i \in A^*$$

and $\sigma(ct_1ct_2\ldots t_{k-1}c) \models \varphi$

The formula $\varphi^\#$ is defined recursively on the structure as follows:

$$m^\# = (c \land XFc) \land (X\psi'_m)$$

where $\psi_m$ is the formula in $LTL_A$ describing $h^{-1}(m) \cap A^*$ and $\psi'_m$ is its relativization
Lifting the formula for $K$

We show that for any formula $\varphi$ in $LTL_M$, there is a formula $\varphi^\#$ in $LTL_\Sigma$ such that

$$w \models \varphi^\# \iff w = ct_1ct_2c\ldots t_{k-1}ct_k, \text{ with } t_i \in A^*$$
and
$$\sigma(ct_1ct_2\ldots t_{k-1}c) \models \varphi$$

The formula $\varphi^\#$ is defined recursively on the structure as follows:

$$m^\# = (c \land XFc) \land (X\psi'_m)$$
where $\psi_m$ is the formula in $LTL_A$ describing $h^{-1}(m) \cap A^*$ and $\psi'_m$ is its relativization

$$(\varphi_1 \land \varphi_2)^\# = \varphi_1^\# \land \varphi_2^\#$$
Lifting the formula for $K$

We show that for any formula $\varphi$ in $LTL_M$, there is a formula $\varphi^\#$ in $LTL_\Sigma$ such that

$$w \models \varphi^\# \iff w = ct_1ct_2c \ldots t_{k-1}ct_k, \text{ with } t_i \in A^*$$

and $\sigma(ct_1ct_2\ldots t_{k-1}c) \models \varphi$

The formula $\varphi^\#$ is defined recursively on the structure as follows:

$$m^\# = (c \land XFc) \land (X\psi'_m)$$

where $\psi_m$ is the formula in $LTL_A$ describing $h^{-1}(m) \cap A^*$ and $\psi'_m$ is its relativization

$$\varphi_1^\# \land \varphi_2^\#$$

$$\neg \varphi^\# = \neg(\varphi^\#) \land (c \land XFc)$$
Lifting the formula for $K$

We show that for any formula $\varphi$ in $LTL_M$, there is a formula $\varphi^\#$ in $LTL_\Sigma$ such that

$$ w \models \varphi^\# \iff w = ct_1ct_2c\ldots t_{k-1}ct_k, \text{ with } t_i \in A^* $$
and $\sigma(ct_1ct_2\ldots t_{k-1}c) \models \varphi$

The formula $\varphi^\#$ is defined recursively on the structure as follows:

$$ m^\# = (c \land XFc) \land (X\psi'_m) $$
where $\psi_m$ is the formula in $LTL_A$ describing $h^{-1}(m) \cap A^*$ and $\psi'_m$ is its relativization

$$ (\varphi_1 \land \varphi_2)^\# = \varphi_1^\# \land \varphi_2^\# $$
$$ (\neg \varphi)^\# = \neg(\varphi^\#) \land (c \land XFc) $$
$$ (X\varphi)^\# = X(\neg cU(c \land \varphi^\#)) $$
Lifting the formula for $K$

We show that for any formula $\varphi$ in $LTL_M$, there is a formula $\varphi^\#$ in $LTL_\Sigma$ such that

$$w \models \varphi^\# \iff w = ct_1ct_2c \ldots t_{k-1}ct_k, \text{ with } t_i \in A^*$$

and $\sigma(ct_1ct_2 \ldots t_{k-1}c) \models \varphi$

The formula $\varphi^\#$ is defined recursively on the structure as follows:

$$m^\# = (c \land XFc) \land (X\psi'_m)$$

where $\psi_m$ is the formula in $LTL_A$ describing $h^{-1}(m) \cap A^*$ and $\psi'_m$ is its relativization

$$(\varphi_1 \land \varphi_2)^\# = \varphi_1^\# \land \varphi_2^\#$$

$$(!\varphi)^\# = !(!\varphi^\#) \land (c \land XFc)$$

$$(X\varphi)^\# = X(!cU(c \land \varphi^\#))$$

$$(\varphi_1U\varphi_2)^\# = (c \implies \varphi_1^\#)U(c \land \varphi_2^\#)$$