

Axioms of Versioning

Marc de Graauw

Intro

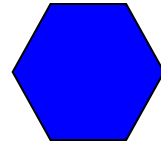
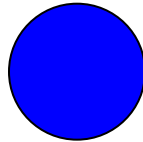
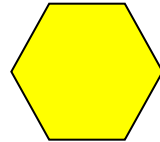
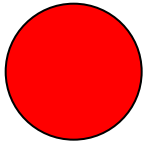
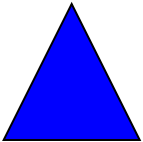
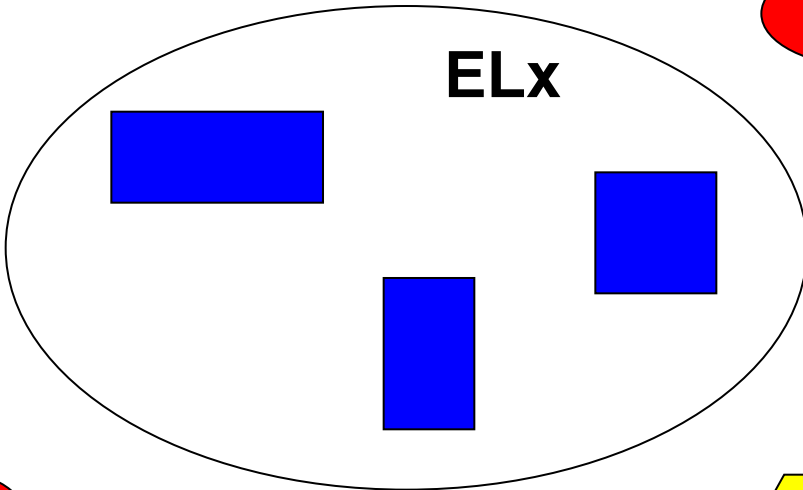
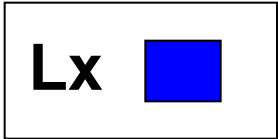
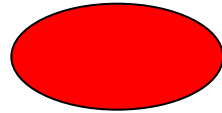
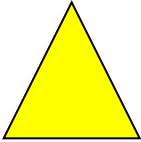
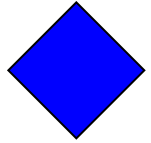
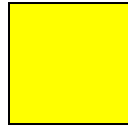
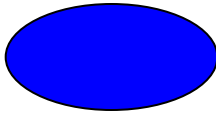
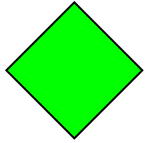
- What is *compatibility* anyway?
- A relation between two: languages?
applications? documents or messages?
schema's?
- What is back- and forward compatibility?
- What does it mean when we say a version
of a language is back- or forward
compatible with another?
- Focus on messaging

Semantical Equivalence Sets

Refreshing Set Theory

- $\{ \text{cat, dog, lion} \}$
- $\{ 1, 2, 3 \}$
- $\{ \text{an integer between zero and four} \}$
- $\{ \} = \emptyset$
- $\{ \text{cat, cat} \} = \{ \text{cat} \}$
- $\{ \text{cat, dog} \} = \{ \text{dog, cat} \}$
- $\{ \text{cat, dog} \}$ subset of $\{ \text{cat, dog, mouse} \}$
- $\{ \text{cat, dog} \}$ intersection $\{ \text{cat, lion} \} = \{ \text{cat} \}$
- $\{ \text{cat, dog} \}$ union $\{ \text{cat, lion} \} = \{ \text{cat, dog, lion} \}$

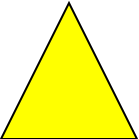
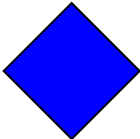
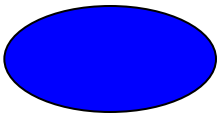
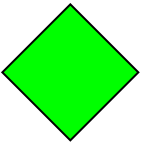
U



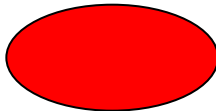
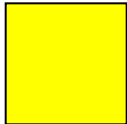
Extension of a language



- There's a language specification, L_x
- L_x documents are blue rectangles
- The *extension* EL_x of L_x are all (possible) blue rectangles
- All non-blues and all non-rectangles are not in EL_x
- The set of all blues is a *superset* of EL_x
- The set of all rectangles are a *superset* of EL_x
- L_y and L_z are extensional superlanguages of L_x

U

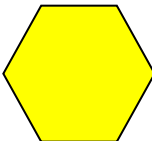
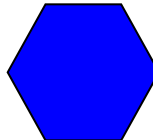
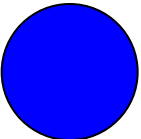
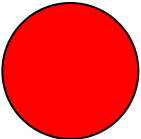
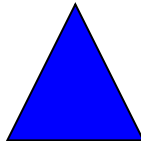
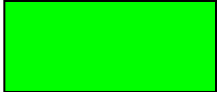
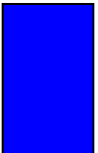
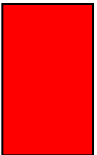
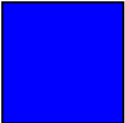


ELy

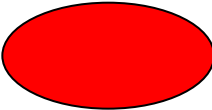
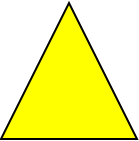
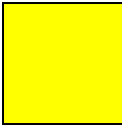
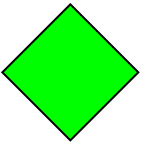


Lx	
Ly	

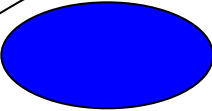
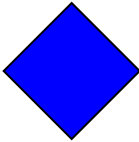
ELx



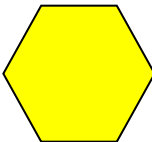
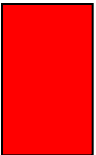
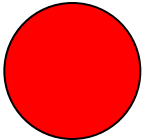
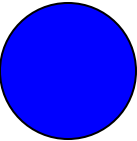
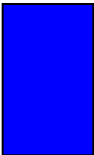
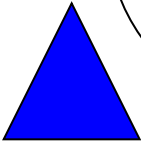
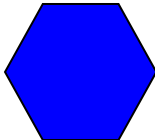
U





ELz






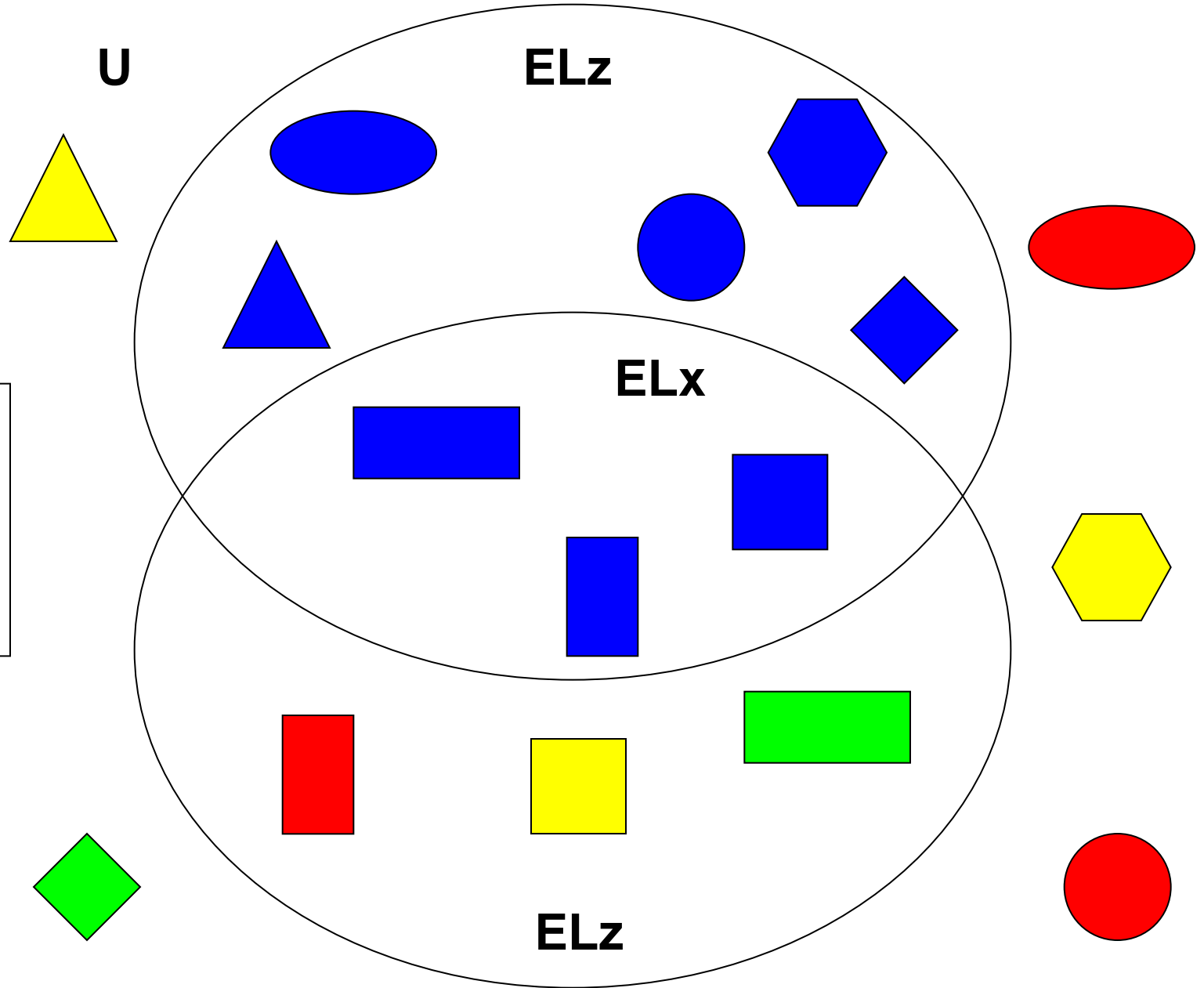
ELx



Lx 

Lz 

Lx	
Ly	
Lz	



Extensional equivalence

- L1: about traffic lights
 - `<code>1</code>` = Red
 - `<code>2</code>` = Green
- L2: about gender
 - `<code>1</code>` = Male
 - `<code>2</code>` = Female

Extensional equivalence

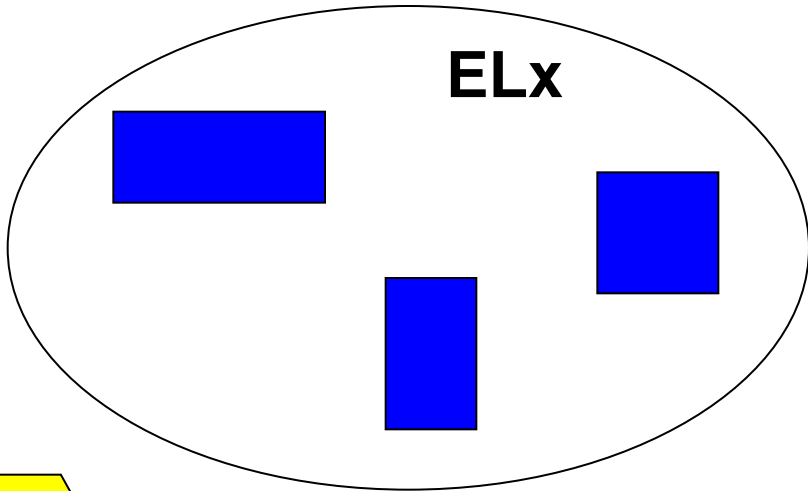
- Extensional equivalence only applies to the set of documents
- If two languages have the same documents, they are extensionally equivalent, even if they are about completely different things
- L1 may be about traffic lights and L2 about gender, but if for both the set of documents is { `1`, `2` } then they are extensionally equivalent, even if their meaning is completely different

U

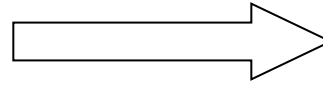
Lx



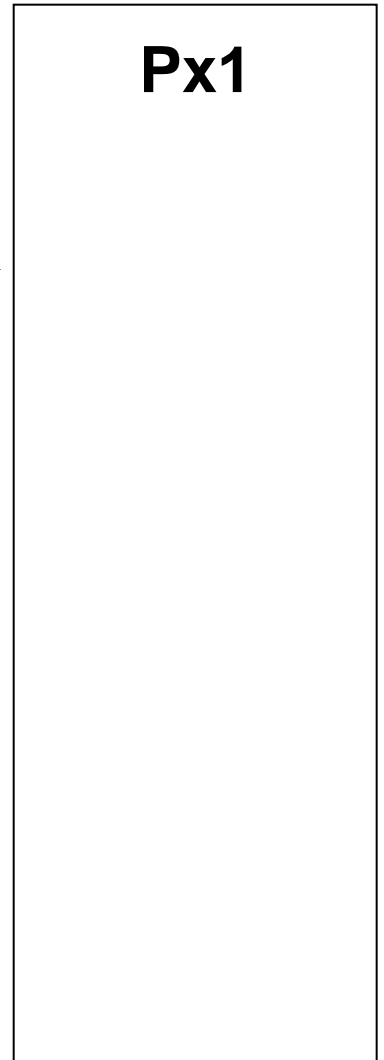
ELx



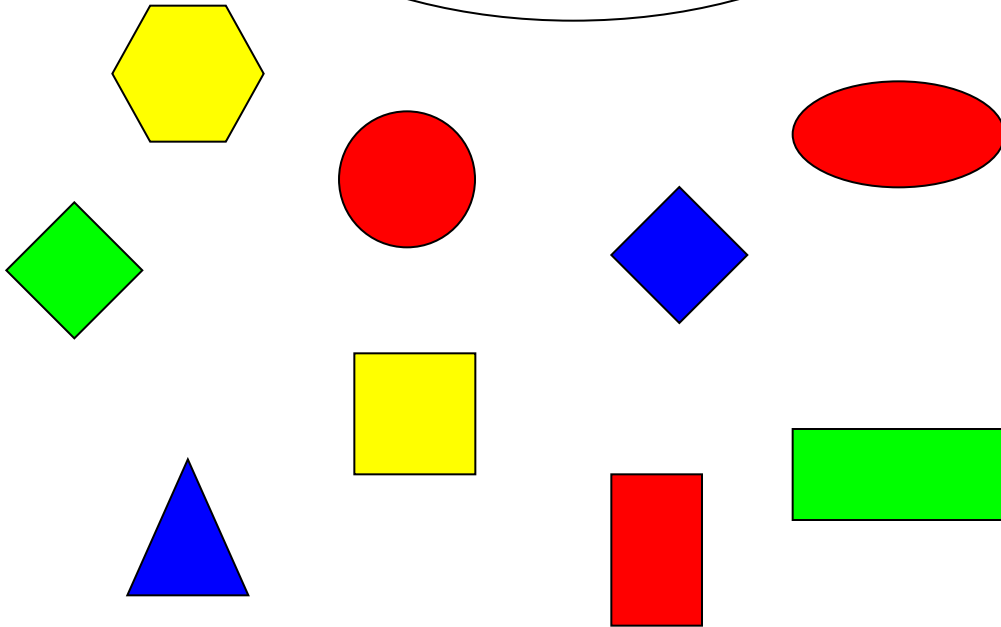
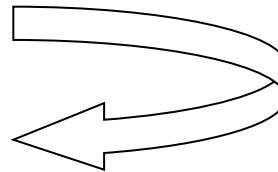
accept



Px1



reject



Accept and Reject

- Applications can read and write (consume/produce, send/receive) documents
- Assume now: docs read == docs written
- Processor P_x accepts all blue rectangles, and rejects all other documents
- Through accepting and rejecting P_x establishes the set of documents which are syntactically conformant with L_x

Semantics of a language

- Some languages may define little semantics
- Most often: in natural language description
- Plus: schema's, UML, other
- Use formal semantics for defining compatibility?
 - logic, OWL
 - too hard for capturing semantics of a complex language as HL7v3
 - defeats the problem

Semantics and Behavior

- Ground notion of compatibility in application behavior?
- Language spec should enable an engineer to implement conforming application
- Application exhibits behavior
- Word processor reads doc
 - behavior: display text, properly formatted
- Medical application receiver prescription message
 - behavior: display appropriate medicine and dosage
- Languages may not define application behavior

Message endpoint



Behavior of message endpoint

- Message endpoint: black box
- Exhibits some behavior
- But: may be no behavior at all
 - change of address message: no immediate visible behavior
- Non-deterministic: pharmacist may ignore or change medical prescription

```
<code code="27"  
  codeSystem="2.16.840.1.113883.2  
  .4.4.5" />
```

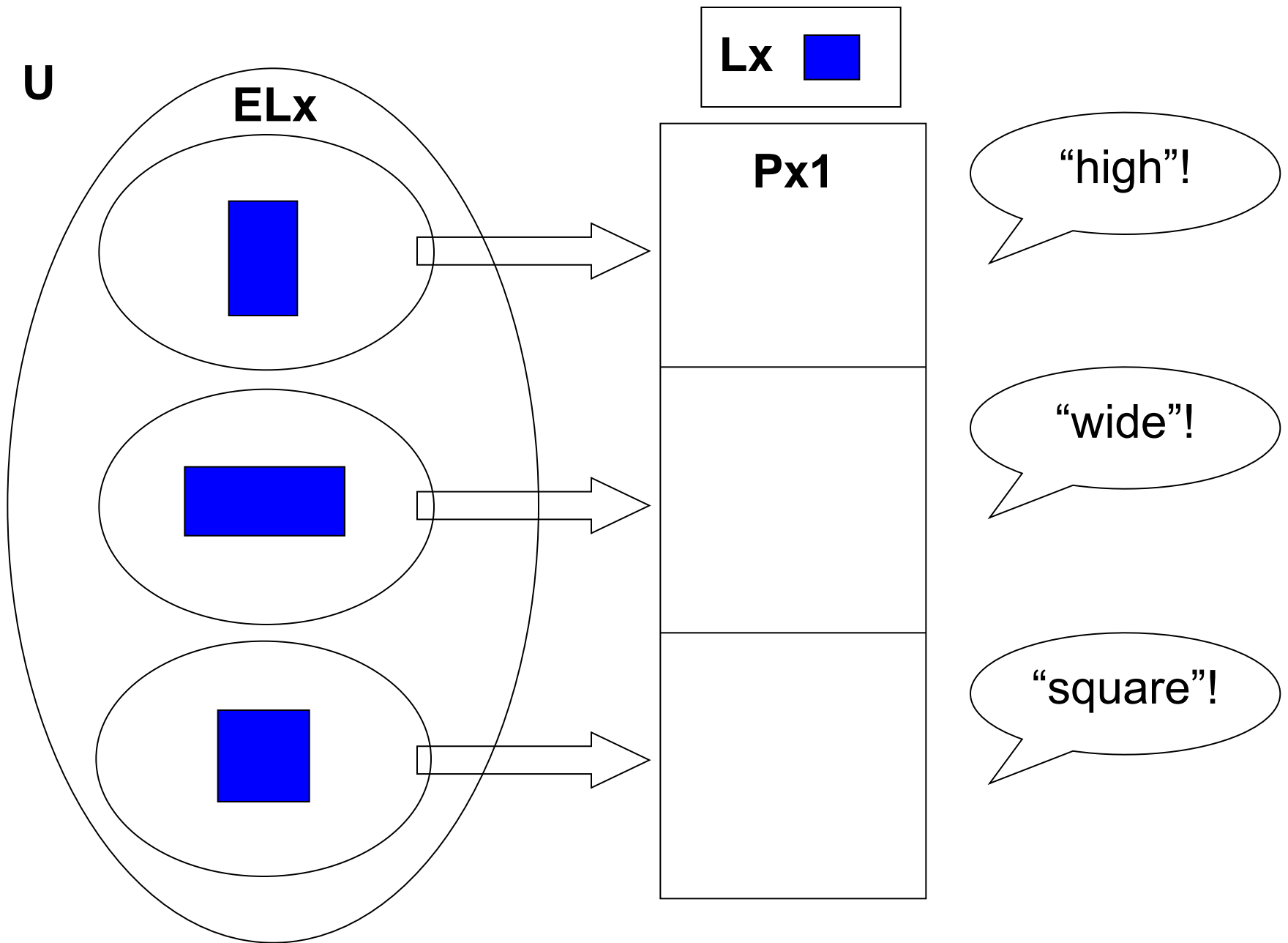
"Dissolve in water"

Message endpoint



Behavior of message endpoint

- Endpoint is black box
- Take humans “out of the box”
- Humans judge behavior
- Assume deterministic behavior
 - start-state plus message determine end-state
- Behavior of endpoint is testable condition
- Real life behavior does not matter much
- Base semantics in behavior: perfect fit





Processors and Behavior

- For every document received, Px exhibits behavior
- Does document determine behavior?
- Reliable messaging
 - order is executed
 - duplicate is ignored
- Document determines behavior
 - assuming the same start-state
 - enough to make compatibility testable

Compatibility is a relation between two applications

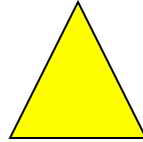
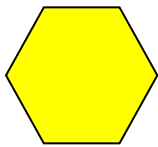
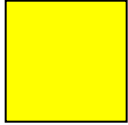
Some Idealizations



- Every language specification is flawless
- There is a perfect processor for every language
- That processor is an exemplar of the language
- Not: docs read == docs written

Lx 


Px

U



Ly 


Py

make

reject

accept

reject

ELy

MLx

accept

reject

ELx



MLy

make



reject

Syntactical Compatibility

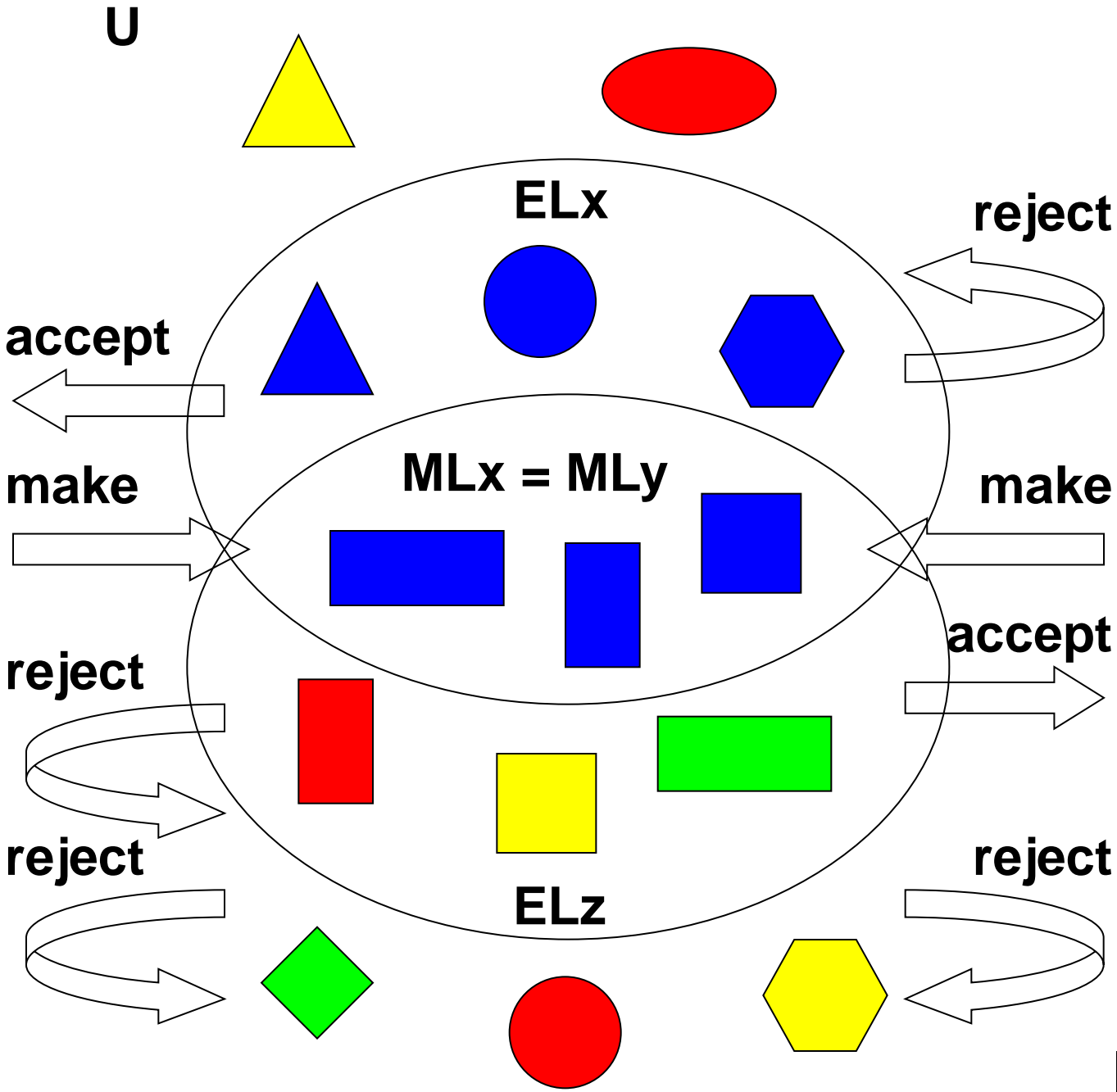
- Two language processors are syntactically compatible if they accept each other's documents
- Common case: both implement the same language
- Also: one processor makes orders, second processor accepts those but only makes order confirmations

Lx 


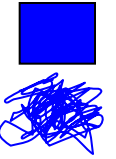
Px

Ly 


Py




Lx

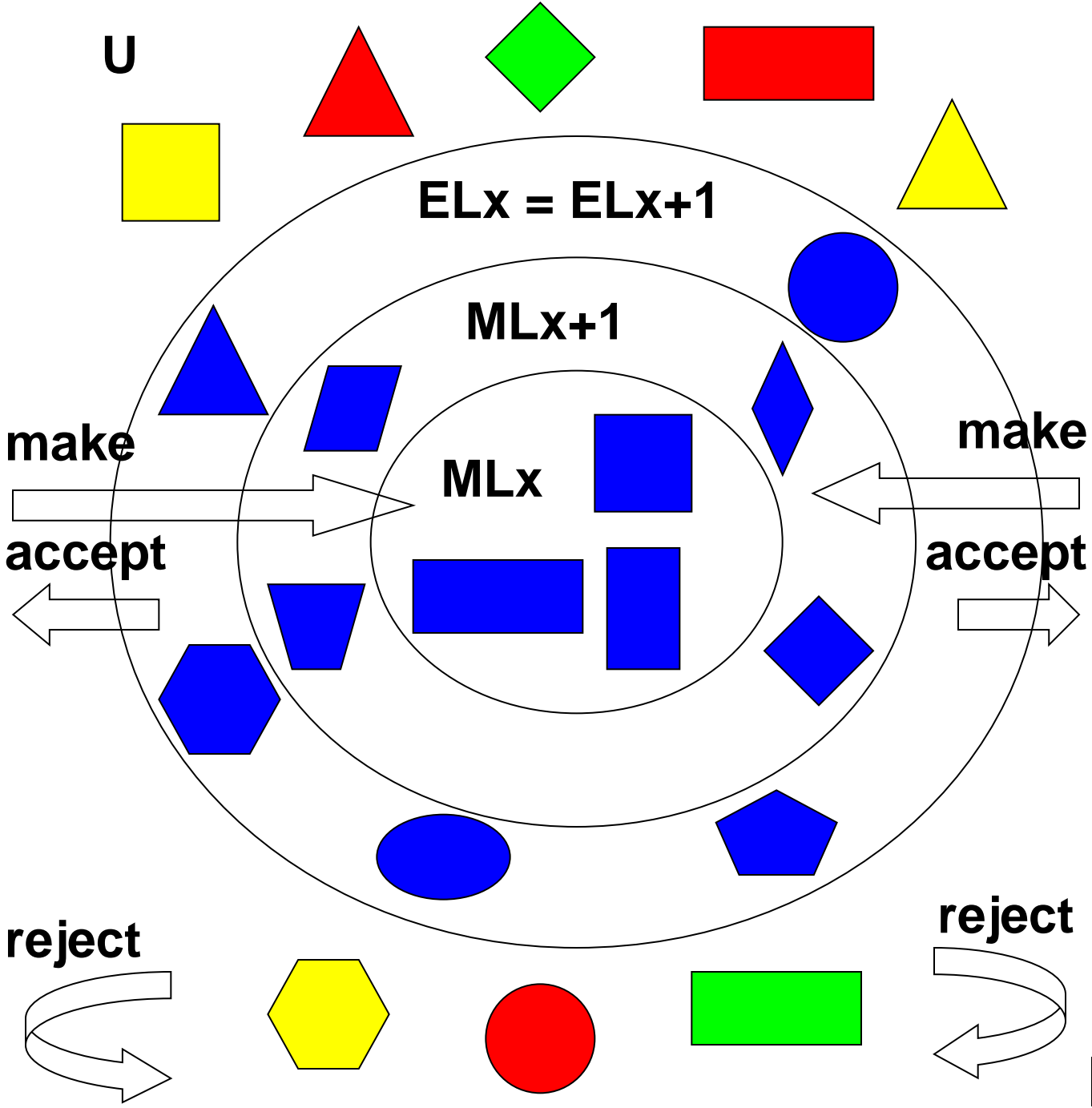


Px

Lx+1

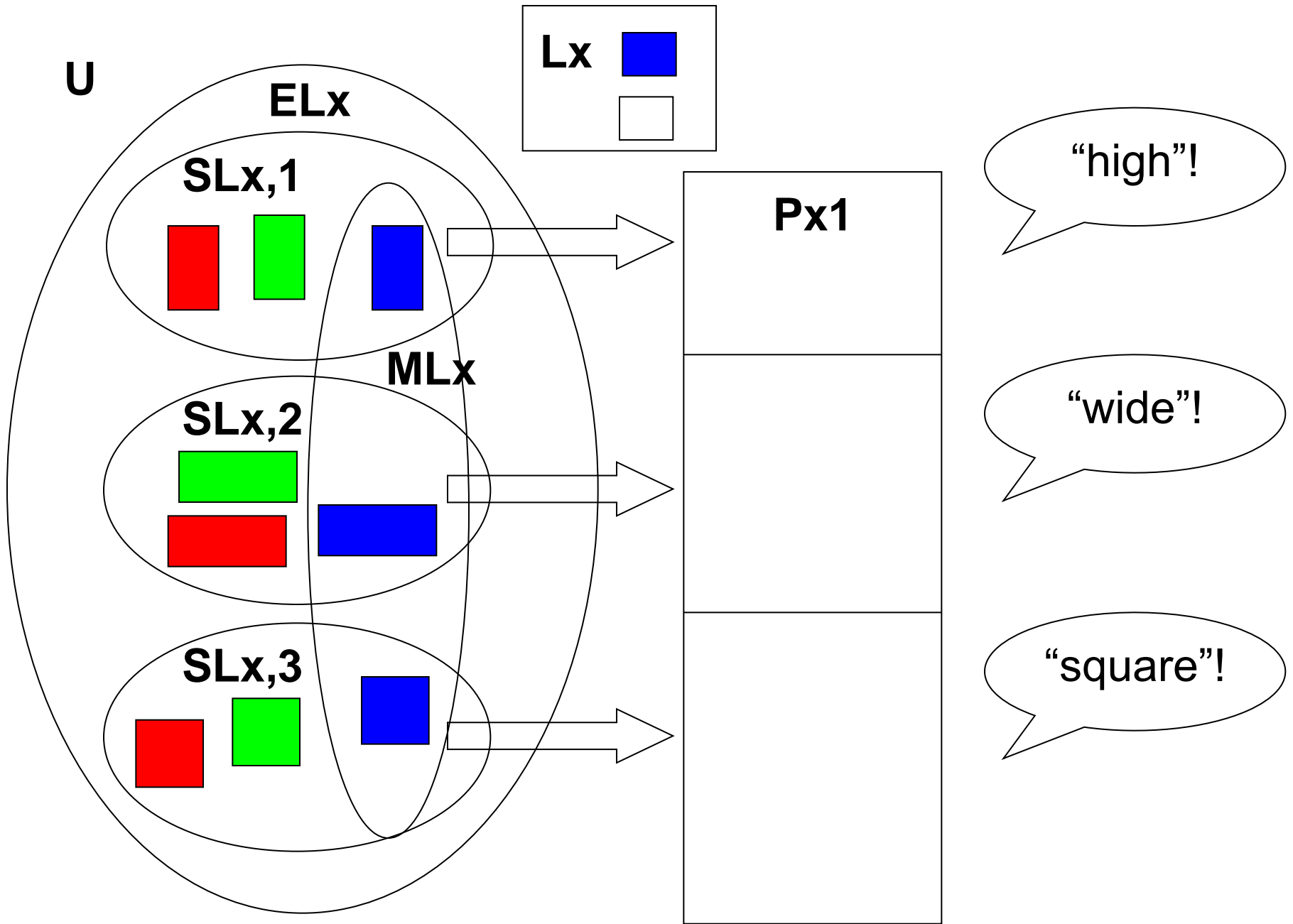


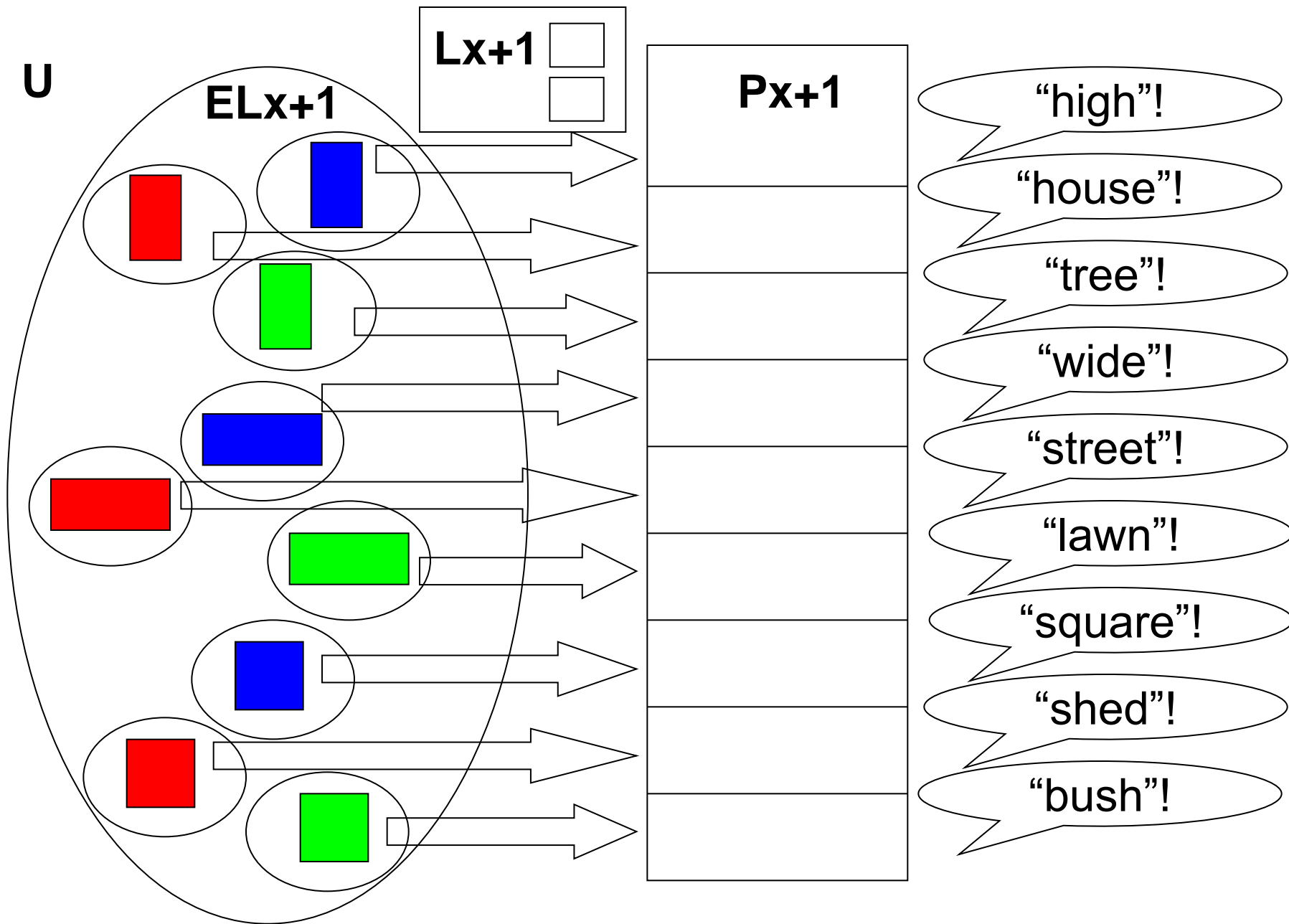
Px+1



Compatibility and Versioning

- Backward syntactical compatibility: P_{x+1} must accept all documents of P_x
- Forward syntactical compatibility: P_x must accept all documents of P_{x+1}
- Key: make a processor accept more than it produces (or understands)
- Accept/reject establishes syntactical compatibility
- Accept/reject is itself behavioral (thus semantical)





Semantical Equivalence Set

- For each document d which P_x accepts, there is a set of documents, with which P_x does the same as with d : the semantical equivalence set $SL_{x,d}$ of document d
- P_{x+1} slices $SL_{x,d}$ up into smaller sets, for which P_{x+1} adds behavior

Semantical Equivalence Set

- For every document produced by P_x , P_{x+1} will behave as P_x expects
 - More general: P_{x+1} should not violate the semantics defined in L_x
- It is safe for P_x to send messages to P_{x+1}
- For documents produced by P_{x+1} , P_{x+1} will 'know' how P_x will behave
 - If P_x 's behavior is not acceptable, P_{x+1} should not send the message
 - More general: P_{x+1} must ensure P_x will not accept the message and exhibit the behaviour defined in L_x
- It is safe for P_x to send messages to P_{x+1}

Implementing
syntactical forward compatibility
enables
semantic enhancement

Conclusion

The basics of compatible versioning are:

1. make sure P_x accepts more documents than it produces (or fully understands), and,
2. partition the semantical equivalence sets of P_x into smaller, more refined semantical equivalence sets for P_{x+1} with additional (new) behavior.

Definitions

1. Two languages are *extensionally equivalent* if they accept the same set of documents.
2. A language is an *extensional sublanguage* of a second language if all documents accepted by the first language are also accepted by the second.
3. If two processors behave the same for every document which belongs to a language L_x , the processors are *behaviourally equivalent* for L_x .
4. Two languages are *syntactically compatible* if they accept the documents produced by each other.
5. A language change is *syntactically backward compatible* if a new receiver accepts all documents produced by an older sender.
6. A language change is *syntactically forward compatible* if an old receiver accepts all documents produced by a new sender.
7. If two languages take the same documents as input, and their processors behave the same for every document, the languages are *semantically equivalent*.
8. The *semantical equivalence set* of a document d is the set of documents which make a processor behave the same as d .
9. A language is a *semantical superlanguage* of a second language if for all documents produced by the second language, processors of the first language behave the same as processors of the second language.
10. A later version of a language is *semantically backward compatible* with an earlier version if the later version is a semantical superlanguage of the earlier one (an old sender may expect a newer, but semantically backward compatible, receiver to behave as the sender intended).
11. An earlier language is *semantically forward compatible* with a later language iff the later language is a semantical sublanguage of the earlier one (this is only possible if a language loses semantics).
12. A later language *semantically extends* an earlier language if the later language introduces new behaviour for some documents accepted, but not produced by the earlier one.