libSBML

New conversion API

Sarah Keating

Frank Bergmann

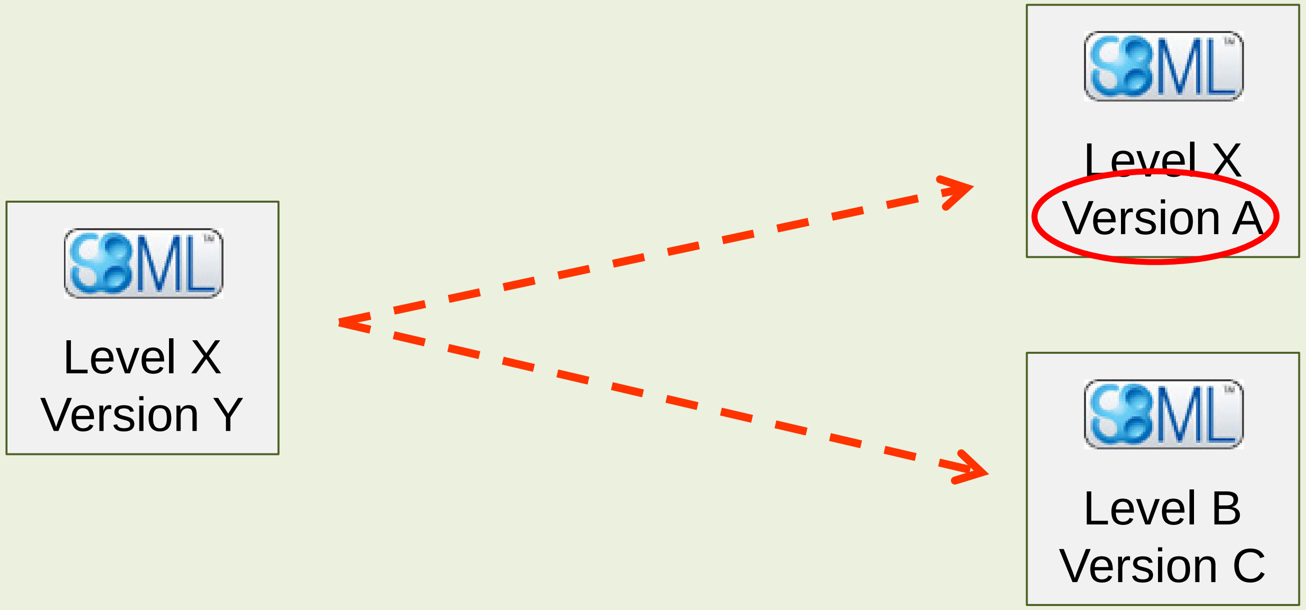
on behalf of the

SBML Team

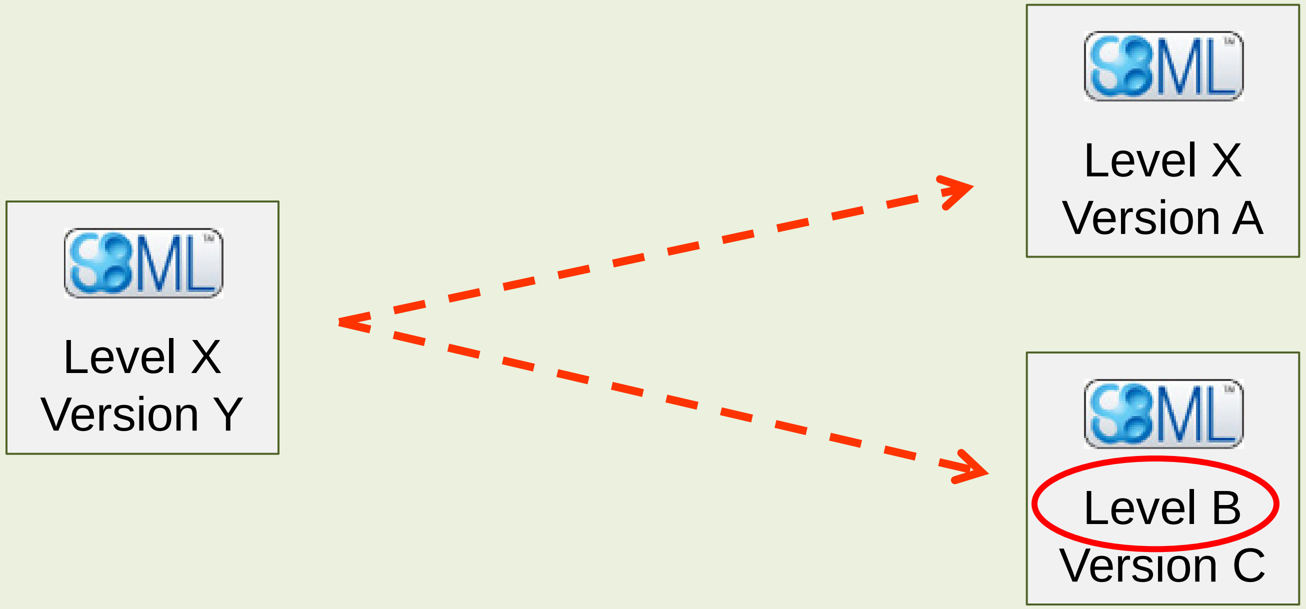
Before packages ...

Level X
Version Y

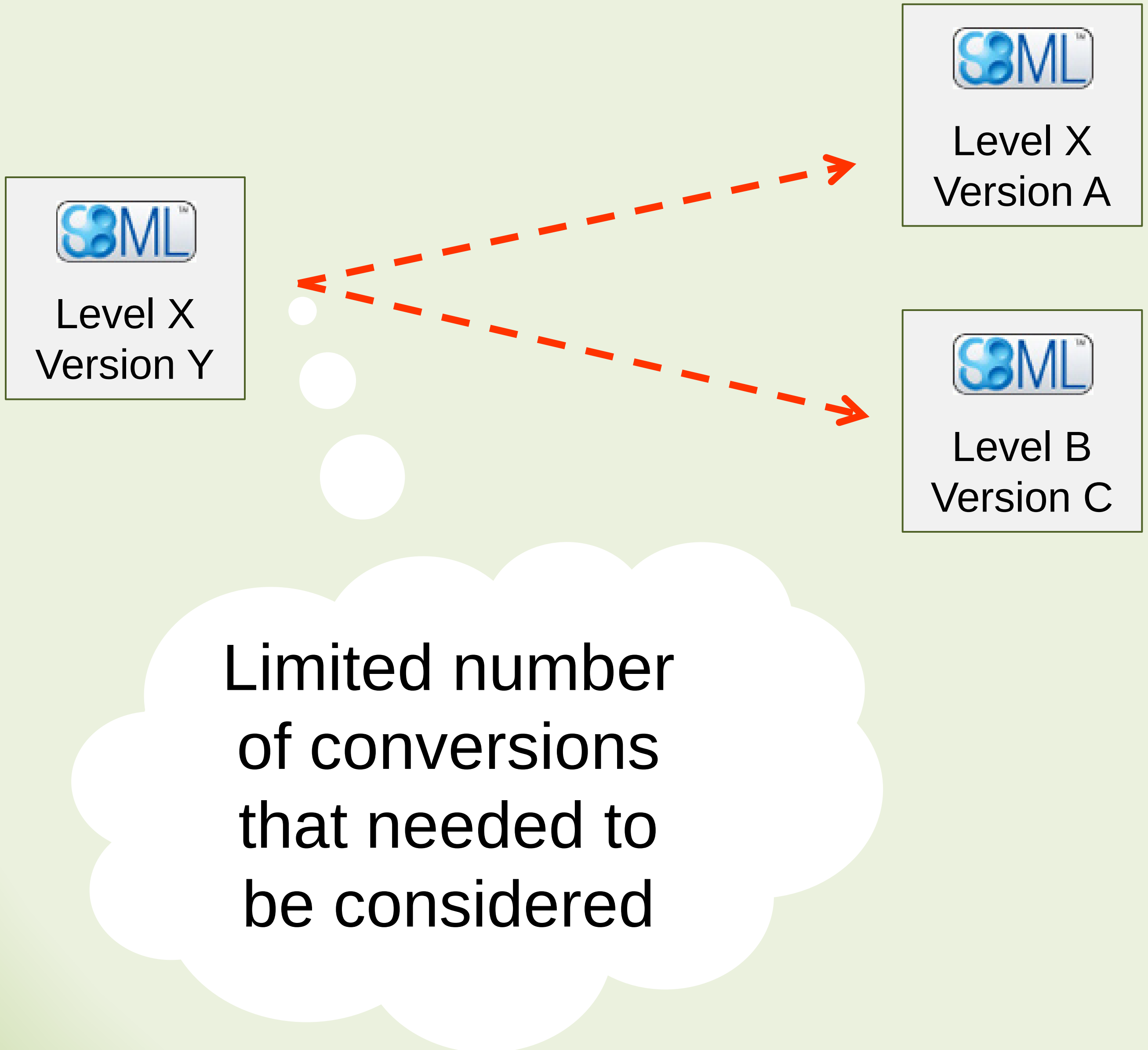
Before packages …



Before packages …

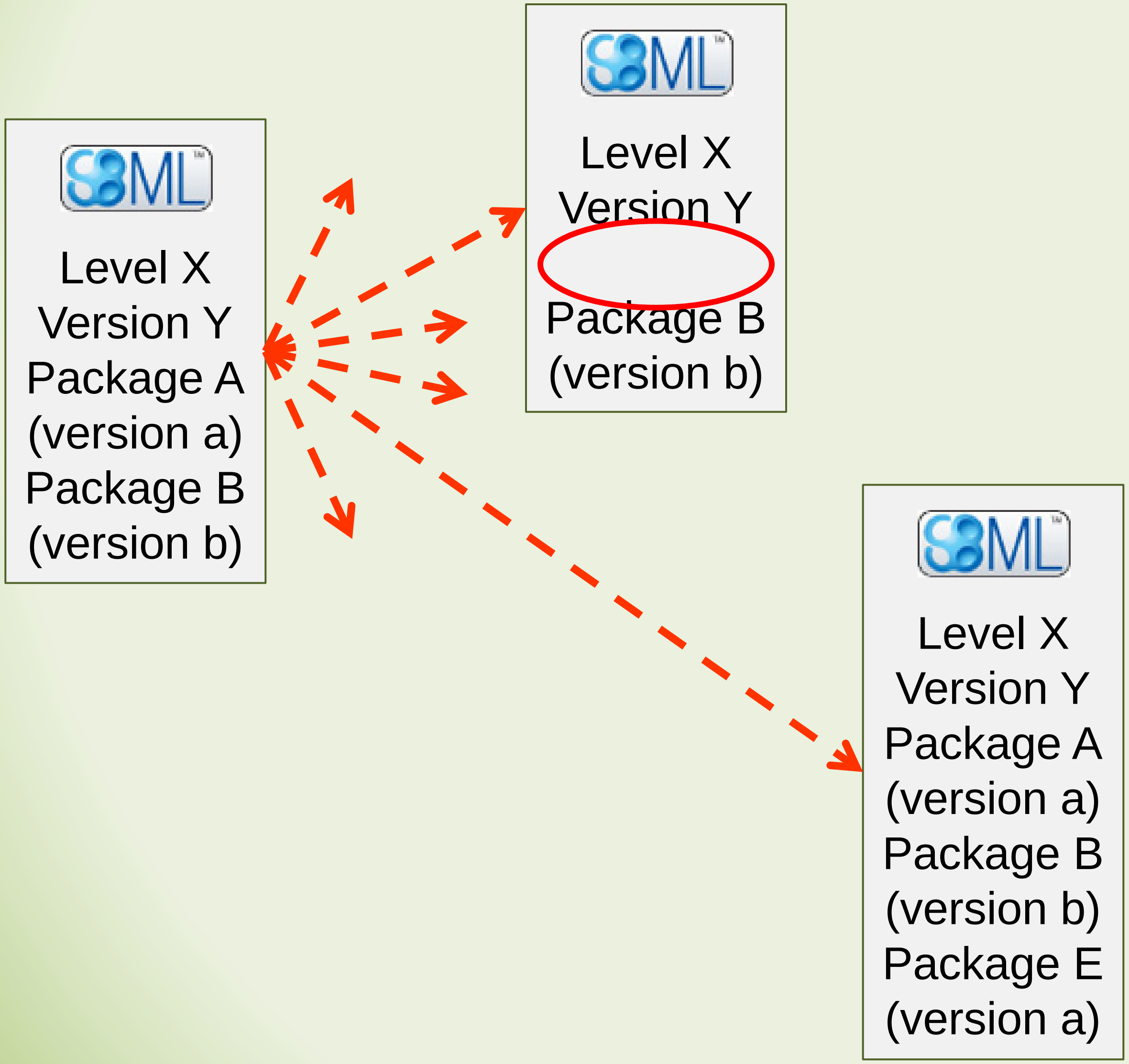


Before packages ...

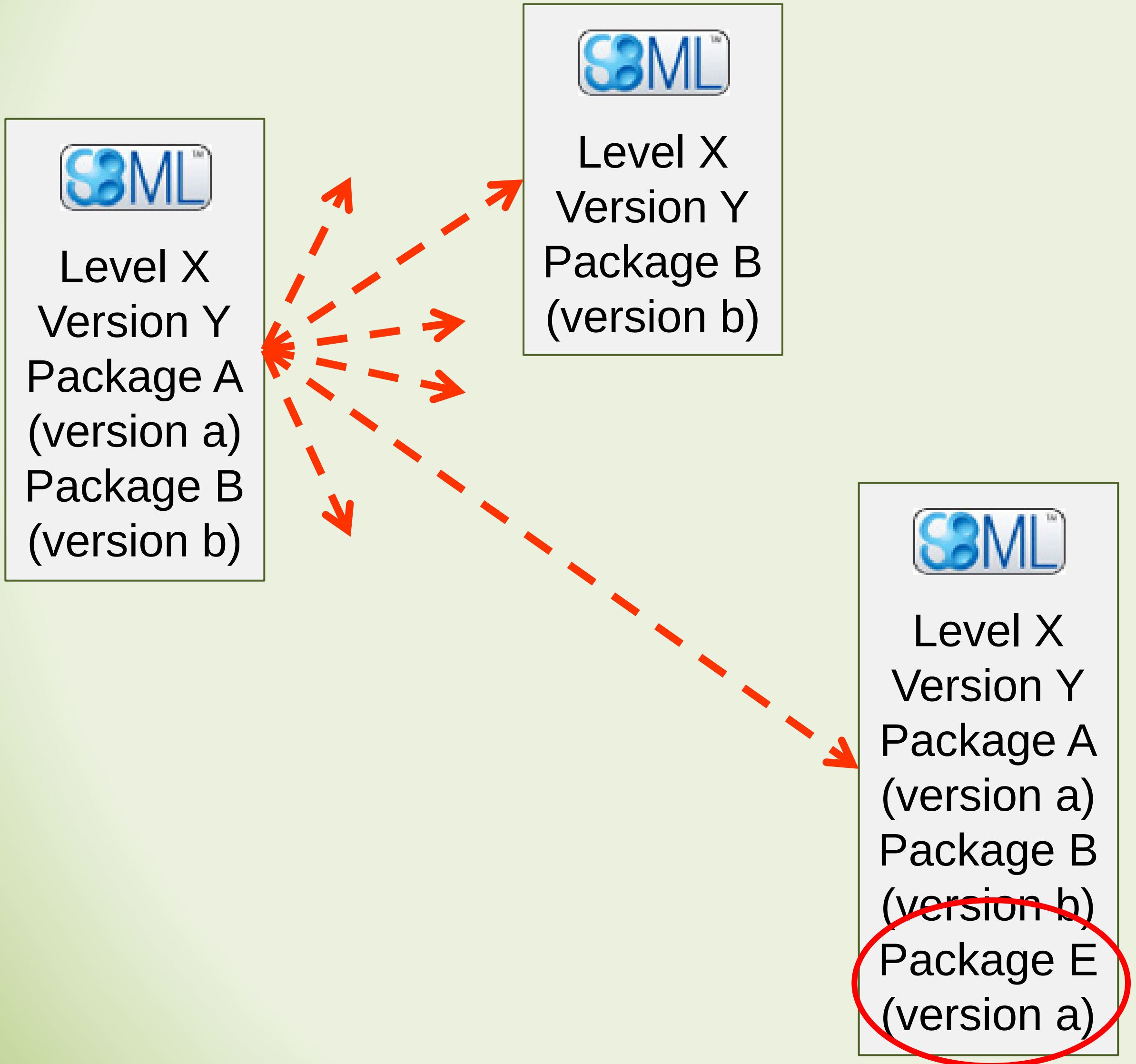


Limited number of conversions that needed to be considered

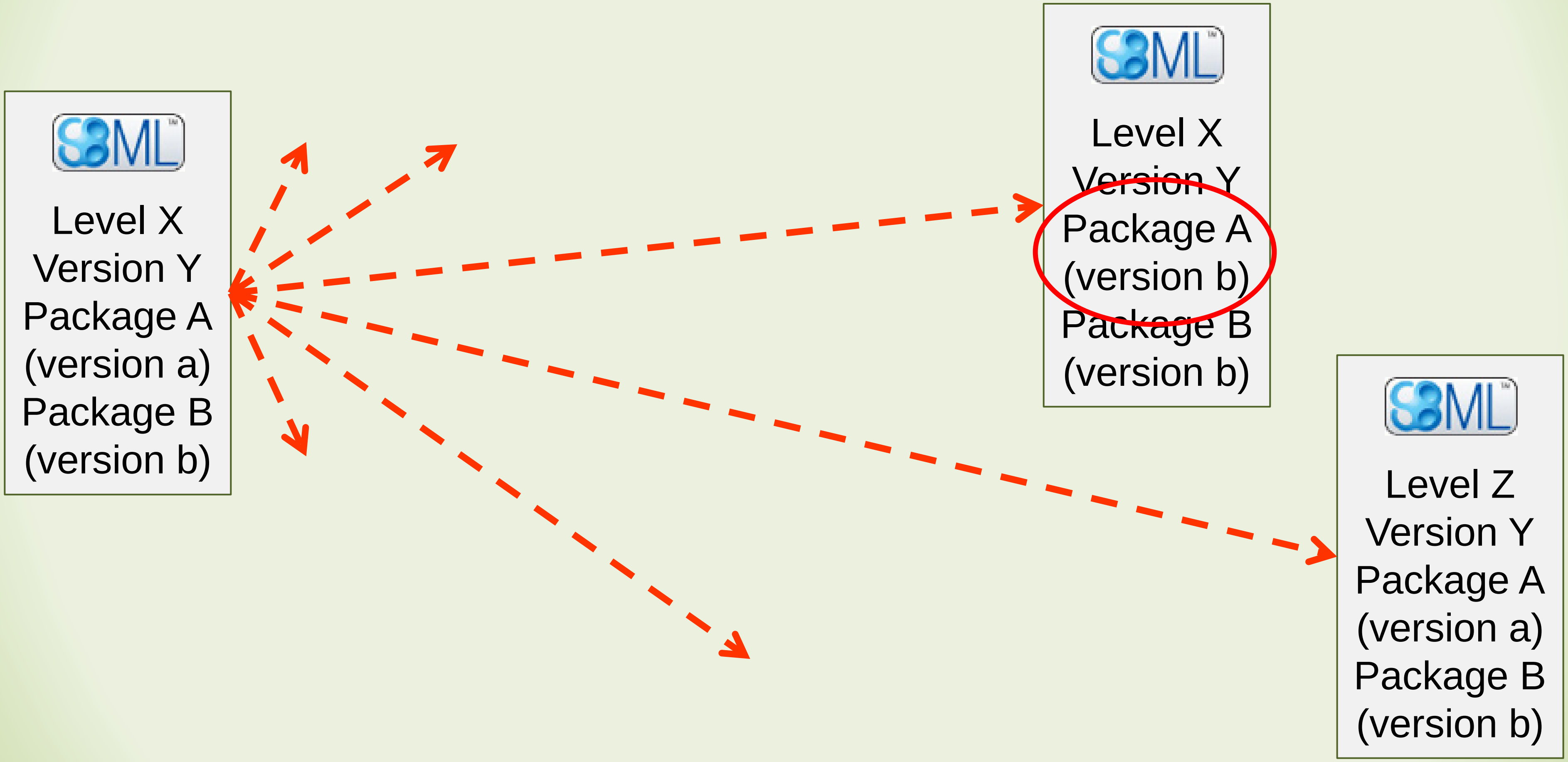
With packages



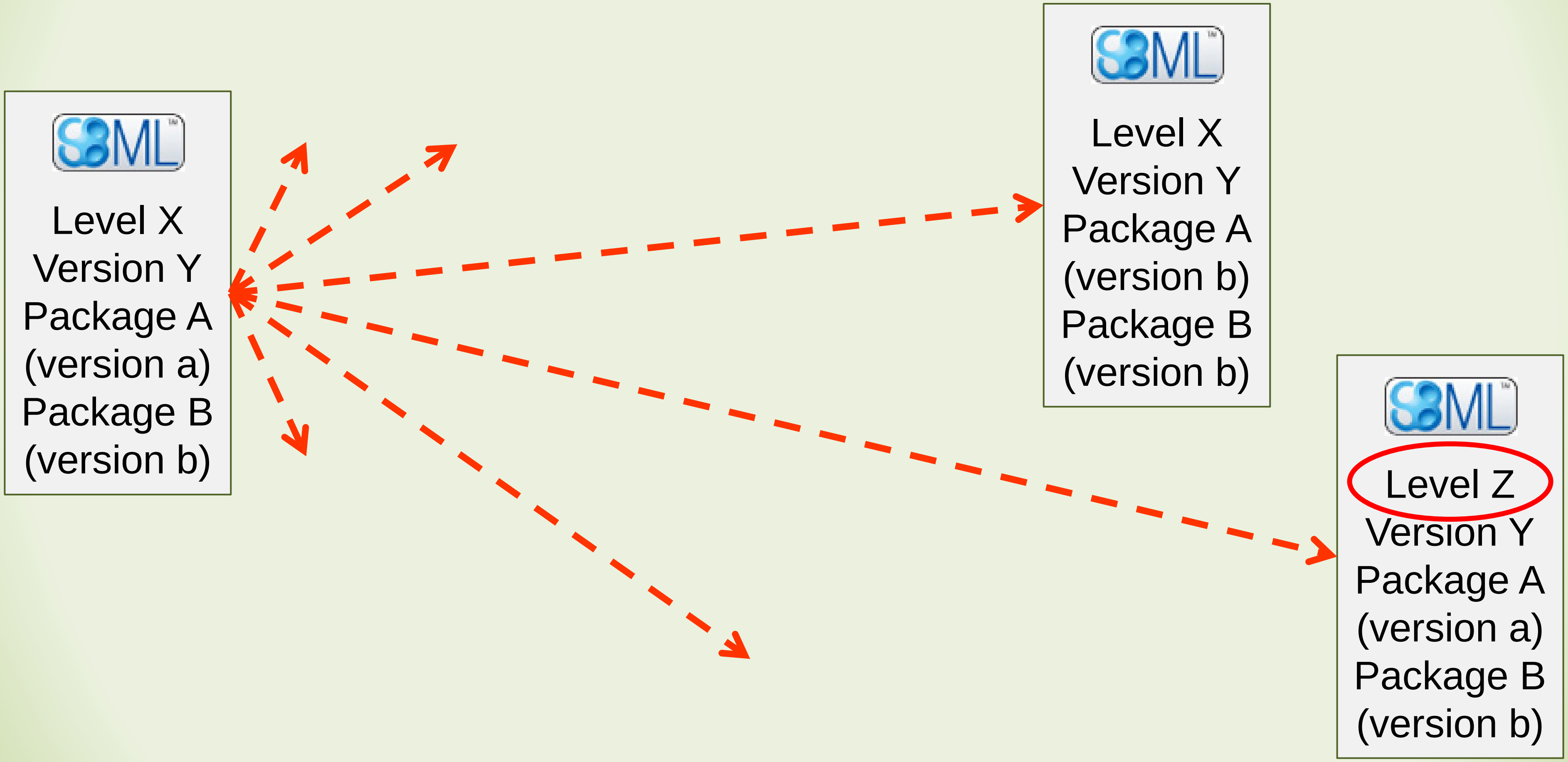
With packages



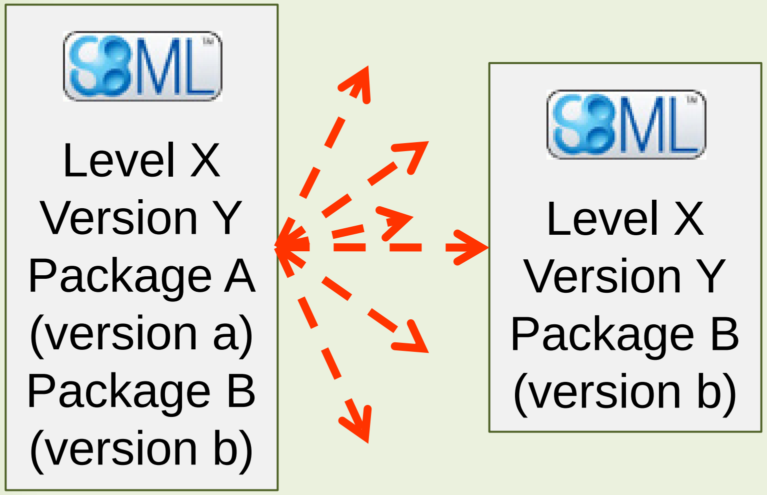
With packages



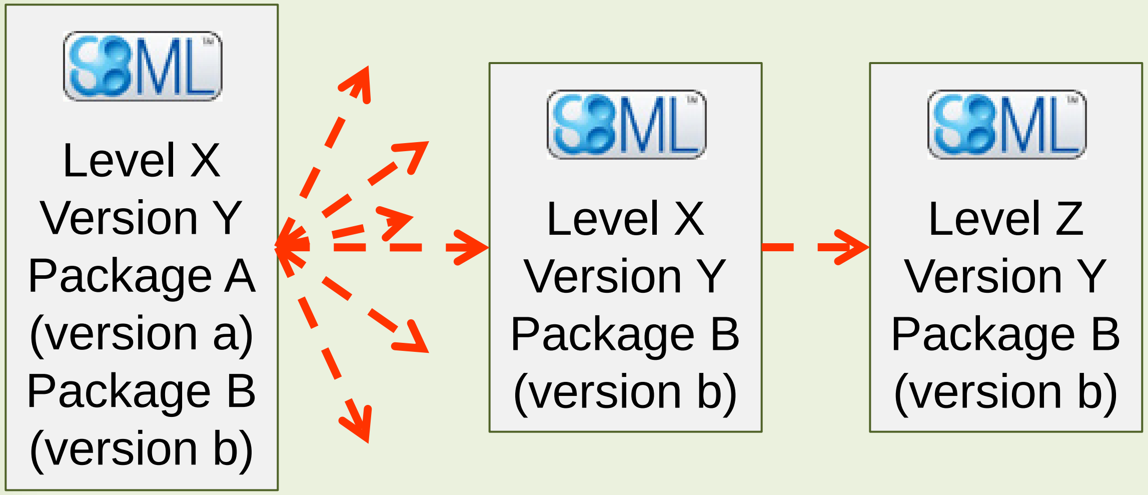
With packages



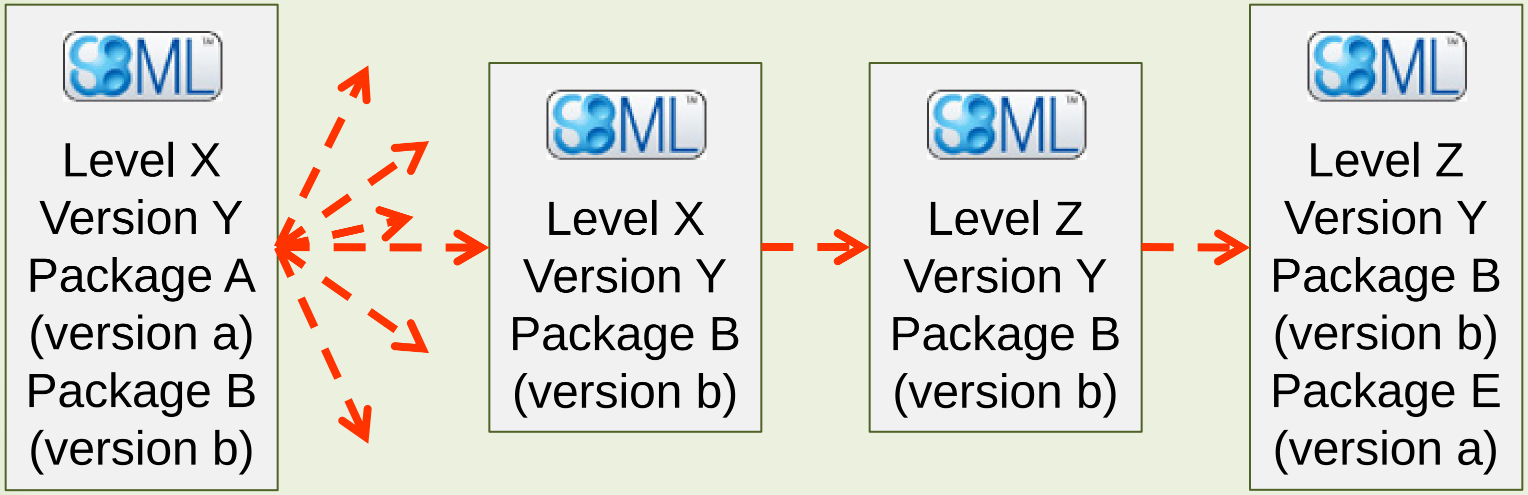
With packages



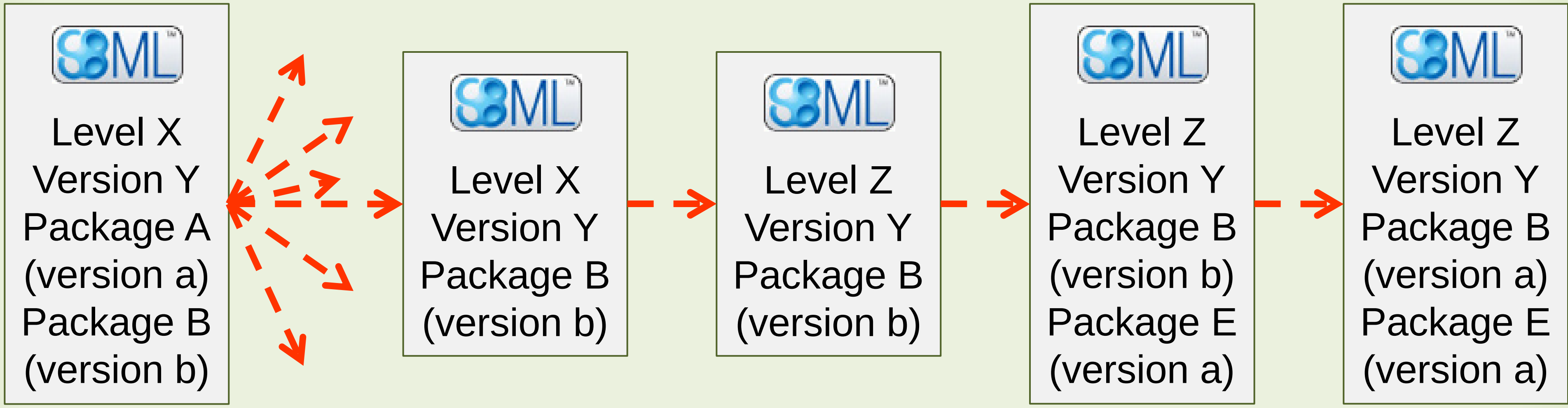
With packages



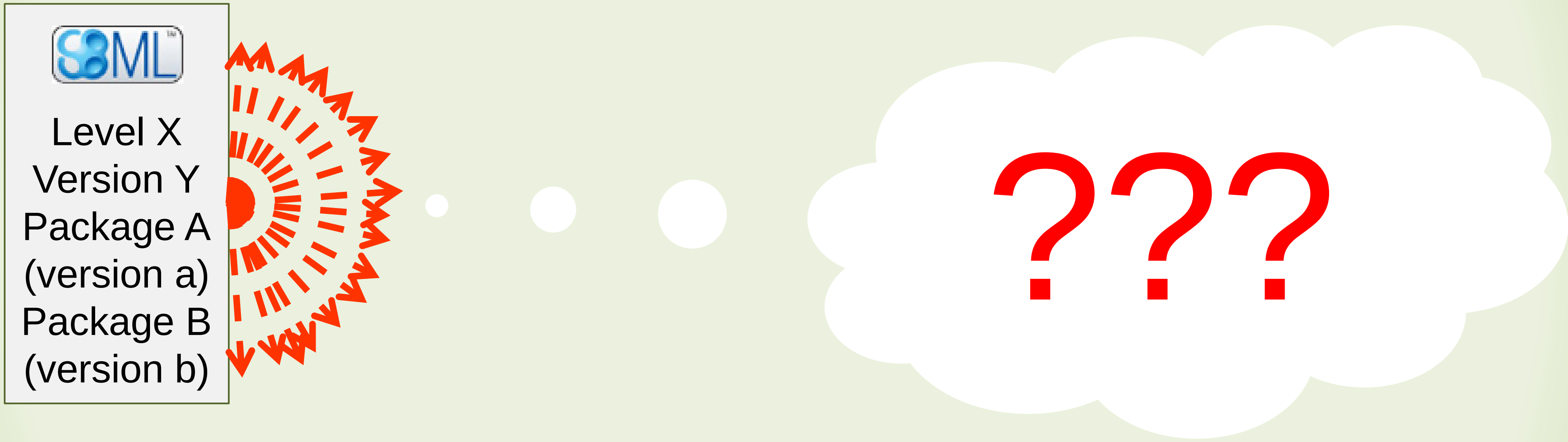
With packages


Level X Version Y Package A (version a) Package B (version b)
Level X Version Y Package B (version b)
Level Z Version Y Package B (version b)
Level Z Version Y Package B (version b) Package E (version a)

With packages


Level X Version Y Package A (version a) Package B (version b)
Level X Version Y Package B (version b)
Level Z Version Y Package B (version b)
Level Z Version Y Package B (version b) Package E (version a)
Level Z Version Y Package B (version a) Package E (version a)

With packages



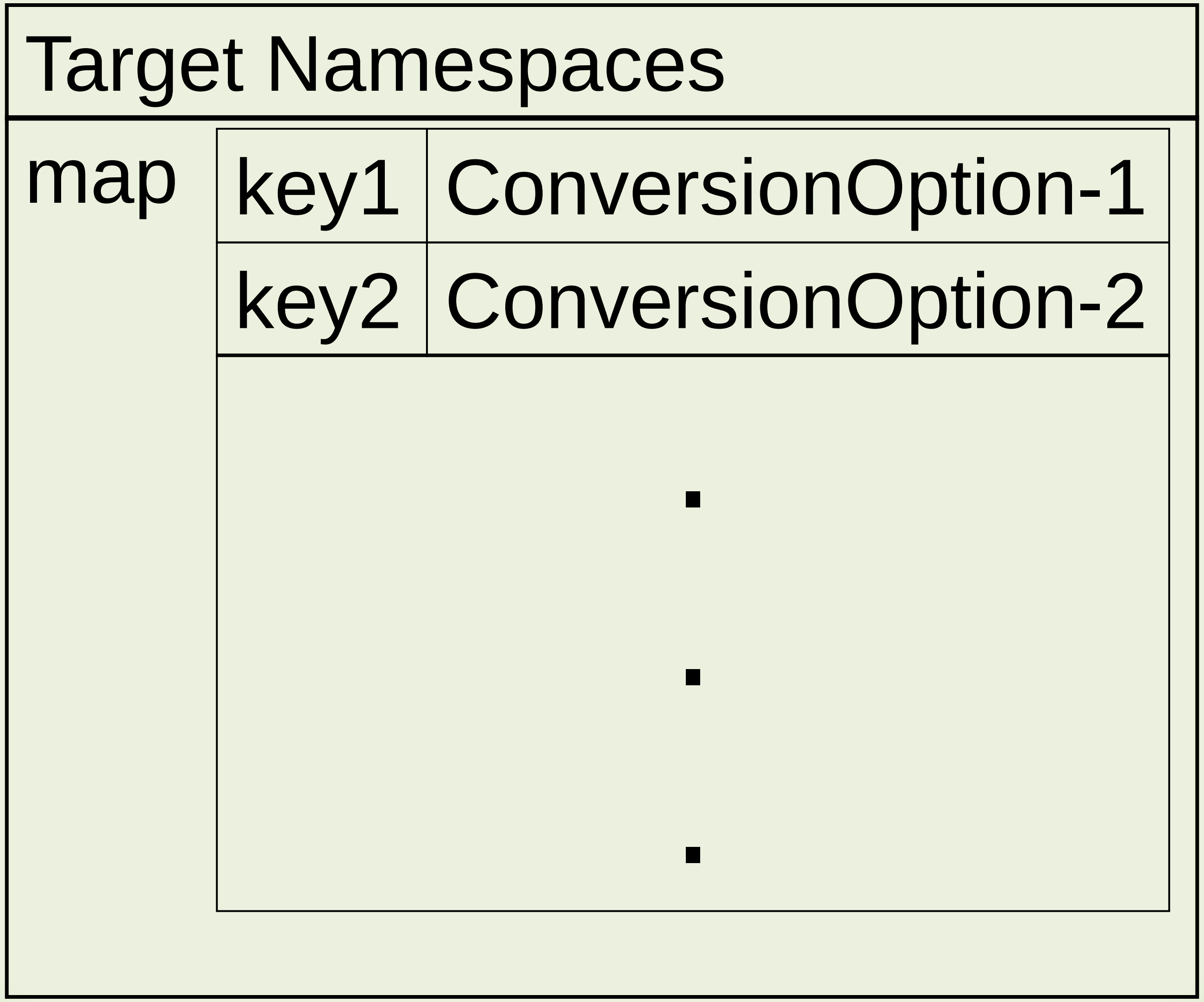
Converters

- provided with libSBML
- provided with packages
- create your own

Registry of converters

access the one you need

Conversion Properties



Conversion Option

key	value	type	description

Conversion Option

key	value	type	description

key

“strict”

Conversion Option

key	value	type	description

key	“strict”
value	true

Conversion Option

key	value	type	description

key	“strict”
value	true
type	CNV_TYPE_BOOL

Conversion Option

key	value	type	description

key	“strict”
value	true
type	CNV_TYPE_BOOL
description	“should validity be preserved”

Conversion Option

key	value	type	description

key	“setLevelAndVersion”
value	true
type	CNV_TYPE_BOOL
description	“change the level and version of the document”

Converters available

- with libSBML-5.1.0-b0
  - setLevelAndVersion
  - expandFunctionDefinitions
  - expandInitialAssignments
  - stripPackage
  - units
- with comp-5.1.0-beta-1
  - flatten comp

BUILDING YOUR OWN CONVERTER

Creating your own Converter

- Inherit from SBMLConverter
- Implement:
  - `Assignment operator / Copy constructor`
  - `virtual SBMLConverter* clone() const;`
  - `virtual ConversionProperties getDefaultProperties() const;`
  - `virtual bool matchesProperties(const ConversionProperties &props) const;`
  - `virtual int convert();`
- Register with registry

Constructors / Operator / Clone

```
SBMLInitialAssignmentConverter::SBMLInitialAssignmentConverter()
: SBMLConverter() { }

SBMLInitialAssignmentConverter::SBMLInitialAssignmentConverter(
const SBMLInitialAssignmentConverter& orig)  :
SBMLConverter(orig) { }

SBMLConverter*
SBMLInitialAssignmentConverter::clone() const {
  return new SBMLInitialAssignmentConverter(*this); }

void SBMLInitialAssignmentConverter::init() {
  SBMLConverterRegistry::getInstance().addConverter(new
SBMLInitialAssignmentConverter());
}
```

getDefaultProperties

```
ConversionProperties
SBMLInitialAssignmentConverter::getDefaultProperties()
const
{
  static ConversionProperties prop;
  prop.addOption(
      "expandInitialAssignments",
      true,
      "expand initial assignments");
  return prop;
}
```

matchesProperties

```
bool
SBMLInitialAssignmentConverter::matchesProperties(const
ConversionProperties &props) const
{
  if ( &props == NULL ||
      !props.hasOption("expandInitialAssignments"))
      return false;
  return true;
}
```

convert

```
int
SBMLInitialAssignmentConverter::convert()
{
  if (mDocument == NULL) return LIBSBML_INVALID_OBJECT;
  Model* mModel = mDocument->getModel();
  if (mModel == NULL) return LIBSBML_INVALID_OBJECT;

  bool success = false;
  /* if no initial assignments bail now */
  if (mModel->getNumInitialAssignments() == 0)
  {
    return true;
  }

  [actual conversion stuff here … ]
  success = (mModel->getNumInitialAssignments() == 0);


  if (success) return LIBSBML_OPERATION_SUCCESS;
  return LIBSBML_OPERATION_FAILED;

}
```

Converter Registry

- Add to registry:

```
SBMLConverterRegistry::getInstance().
    addConverter(new
    SBMLInitialAssignmentConverter());
```

- Or use the register class:

```
static SBMLConverterRegister
<SBMLInitialAssignmentConverter>
registerIAConverter;
```

Calling a known Converter

- Construct ConversionProperties object:

```
ConversionProperties prop(getSBMLNamespaces());
prop.addOption("expandInitialAssignments", true,
"expand initial assignments");
```

- Ask registry for a converter with those properties

```
SBMLConverter* converter =
SBMLConverterRegistry::getInstance().getConverterFor(props);
```

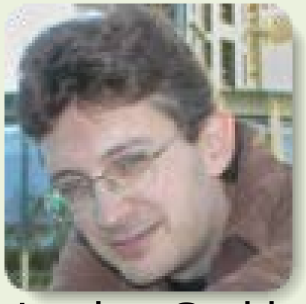
- Apply to document

```
converter->setDocument(this);
converter->setProperties(&props);
int result = converter->convert();
```

SBML Team

Frank Bergmann

Caltech, USA


Lucian Smith

U. of Washington, USA

Sarah Keating

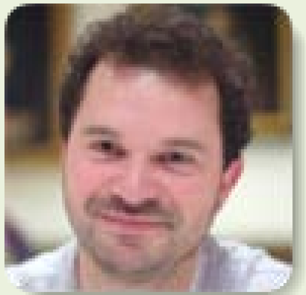
EMBL-EBI, UK

Mike Hucka

Caltech, USA

Linda Taddeo

Caltech, USA


Nicolas Rodriguez

EMBL-EBI, UK

National Institute of General Medical Sciences