Is the XML Schema `xsd:choice` Element Redundant?

Issue

Is `xsd:choice` redundant? Can every `xsd:choice` be replaced by the combination of an abstract element and substitution group?

Replacing `xsd:choice` with an Abstract Element and Substitution Group

Here is an example that uses `xsd:choice`. The content of `transportation` is a choice of `train`, `plane`, or `automobile`:

```
<xsd:element name="transportation">
    <xsd:complexType>
        <xsd:choice>
            <xsd:element name="train" type="xsd:string"/>
            <xsd:element name="plane" type="xsd:string"/>
            <xsd:element name="automobile" type="xsd:string"/>
        </xsd:choice>
    </xsd:complexType>
</xsd:element>
```

Rather than using `xsd:choice`, the schema could be designed using `xsd:sequence` which contains an element that `ref`'s to an abstract element, `mode-of-transportation`. The `train`, `plane`, and `automobile` elements are globally declared and are substitutable with `mode-of-transportation`:

```
<xsd:element name="transportation">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="mode-of-transportation" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="mode-of-transportation" abstract="true"
             type="xsd:string" />

<xsd:element name="train"
             substitutionGroup="mode-of-transportation"/>
<xsd:element name="plane"
             substitutionGroup="mode-of-transportation"/>
```

```
<xsd:element name="automobile"
             substitutionGroup="mode-of-transportation"/>
```

The two designs produce identical results.

Does this mean that the `xsd:choice` element can always be replaced by the combination of an abstract element and substitution group?

Answer: No. There are data designs that can only be expressed with `xsd:choice` and cannot be expressed using an abstract element and substitution group.

Data Designs That Can Only Be Expressed Using xsd:choice

Choice of Sequences

The `xsd:choice` element can be used to express a choice of sequences. Here is an example:

If it is sunny today then I will either:
- go to the beach and then go shopping, or
- go to the movie and then soak in the sauna

It is expressed in XML Schema as follows:

```
<xsd:complexType name="Sunny-Day">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element name="Beach" type="…" />
            <xsd:element name="Shopping" type="…" />
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="Movie" type="…" />
            <xsd:element name="Sauna" type="…" />
        </xsd:sequence>
    </xsd:choice>
</xsd:complexType>
```

A choice of sequences cannot be expressed using abstract element and substitution group.

Any Order

Recall that XML Schemas has the `any` element which says that its content elements can occur in any order. However, `any` can only be used if there is zero or one occurrence of each content element. That restriction can be partially overcome using `xsd:choice`.

Example: The content of `Book` is `Author` followed by one or more `Titles`, or vice versa. This is expressed in XML Schemas as follows:

```
<xsd:complexType name="Book">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element name="Author" type="xsd:string" />
            <xsd:element name="Title" maxOccurs="unbounded"
                         type="xsd:string" />
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="Title" maxOccurs="unbounded"
                         type="xsd:string" />
            <xsd:element name="Author" type="xsd:string" />
        </xsd:sequence>
    </xsd:choice>
</xsd:complexType>
```

This cannot be expressed using abstract element and substitution group.

Here's another example of using `xsd:choice` to express any order. The example comes from XHTML:

The XHTML head element has one `title`, at most one `base`, and zero or more of various other possible children, in any order. The content model is expressed in a choice whose branches are sequences:

```
<xsd:complexType name="head">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element name="title" type="…" />
            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                <xsd:element name="script" type="…" />
                <xsd:element name="style" type="…" />
                <xsd:element name="meta" type="…" />
                <xsd:element name="link" type="…" />
                <xsd:element name="object" type="…" />
            </xsd:choice>
            <xsd:sequence minOccurs="0">
                <xsd:element name="base" type="…" />
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                   <xsd:element name="script" type="…" />
                   <xsd:element name="style" type="…" />
                   <xsd:element name="meta" type="…" />
                   <xsd:element name="link" type="…" />
                   <xsd:element name="object" type="…" />
                </xsd:choice>
            </xsd:sequence>
        </xsd:sequence>
        <xsd:sequence>
```

```
                <xsd:element name="base" type="…" />
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                    <xsd:element name="script" type="…" />
                    <xsd:element name="style" type="…" />
                    <xsd:element name="meta" type="…" />
                    <xsd:element name="link" type="…" />
                    <xsd:element name="object" type="…" />
                </xsd:choice>
                <xsd:element name="title" type="…" />
                <xsd:sequence>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
                       <xsd:element name="script" type="…" />
                       <xsd:element name="style" type="…" />
                       <xsd:element name="meta" type="…" />
                       <xsd:element name="link" type="…" />
                       <xsd:element name="object" type="…" />
                    </xsd:choice>
                </xsd:sequence>
            </xsd:sequence>
        </xsd:choice>
    </xsd:complexType>
```

The same language cannot be defined using substitution groups.

Other Differences

Elements in a substitution group must be declared globally, whereas elements declared in `xsd:choice` may be locally declared. This has several implications:

- Any globally declared element may be used as the root element of an XML instance document (and there is no way to prevent this), whereas local elements cannot.
- Global elements must always be namespace qualified, whereas local elements are not namespace qualified (when `elementFormDefault="unqualified"`).

This is subjective, but many developers find `xsd:choice` to be simpler and easier to understand than substitution groups.

Summary

The `xsd:choice` cannot be replaced by substitution groups in all cases. There are several important data designs that can only be expressed with `xsd:choice` and cannot be expressed with substitution groups:

- choice of sequences
- any order

Acknowledgements

The following people contributed to this paper:

- George Cristian Bina
- Kevin Braun
- Pete Cordell
- Roger Costello
- Andy Davidson
- Daniel Dui
- David Ezell
- Mukul Gandhi
- Michael Kay
- Michael Sperberg-McQueen