SVG XML path compression results

- SVG path@d string as an XML element
  - Each primitive (M, c…) as an element child
  - Each coordinate as a separate attribute
- SVG XML Path example
  - String value
    - M100,200 C100,100 250,100 250,200 S400,300 400,200L800,550 L1000,300
  - XML equivalent
    - (Note: Other designs are possible)

    -
      ```
      <Path>
              <M x="100" y="100"/>
              <C x1="100" y1="100" x2="250" y2="100" x="250" y="200"/>
              <S x1=“400" y1=“300" x=“40" y=“200"/>
              <L x1="1000" y1=“300"/>
      </Path>
      ```
    - Path element has 20 possible sub-children
    - Coordinate values defined as xs:float
- Generalization
  - Same scheme is applicable to several other attributes
    - @transform, @points, @keyTimes
    - animateTransform@values

A practical example

- Results without compression
  - SVG Path@d string
    - 68*8 = 544 bits
  - SVG XML Path
    - 155*8 = 1240 bits
- SVG Path@d compression
  - Restricted character set encoding
  - Each attribute value character encoded as 6 bits codes
    - 68*6 = 408 bits (33%)
- SVG XML Path compression
  - EXI-based encoding
    - Schema informed grammars for Path element and children
    - Each primitive (M, c…) encoded as a priority
      - 20 productions, hence 5 bit
    - Each coordinate encoded using a typed encoding (float or integer)
      - Float = 18 bits at minimum, Integer = 9 bits at minimum
  - Schema in strict mode with float encoding
    - 5*5 + 16*18 = 313 bits (75%)
  - Schema in strict mode with integer encoding
    - 5*5 + 4*9 + 12*17 = 265 bits (105%)
  - Schema in deviation mode with float encoding
    - 5*5 + 5*1 + 16*1 + 16*18 = 334 bits (63%)

XMLized path compression results

- Tests on SVG subset
  - Both long paths and small paths
  - Not meant to be representative for all SVG documents
- Compression results compared to XML
  - Strict mode with all values encoded using EXI float encoding
  - Size divided by a factor of 1.3 to 2.0 compared to current short syntax encoding
- Overall assessment
  - Simple with good results
    - Although not as good as dedicated SVG codecs
      - Both in terms of compression and efficiency
  - Good mapping with SVG DOM Path API
  - Compression benefits from using relative primitives (m, c…)
- Items for further study
  - SVGZ and EXI compression mode performances comparison
  - Schema knowledge sharing
    - Need for schema agreement between decoder and encoder
    - SVG schema is huge and may be difficult to handle for small devices
    - Most compression may be gained by a small schema part
      - Only path data related schema may be shared between encoder and decoder
  - Use of integer to encode coordinates boosts the compression
    - Not usable in every use case
    - May be difficult to apply on per-path scope, better suited to document scope
    - May require several schema definitions or use of data type representation map feature