Formal Specification of Our Extended Vega-Lite

```
Notation
“a := ”: a is defined as b, “a ~ ”: possible names of a is, “...”: extensible, “|”: or,
“(a)”: optional argument, “<Abc>”: datatype, “[]”: a list of, “[0..n]”: a list of length 0 to n,
“(a|b|..)”: either one of a, b, or ...
“{}”: key-value map, “<Number>” means either number or a string with its unit (e.g., 350, “350px”).
+: see related Vega-Lite documentations.
ExVLSpec := (Name), Data, Layout, Layer, (Transform, Interaction, Title, NonData)
Name := <String> the name of a visualization
Data := <JSON> the dataset to generate the chart

Layout := (Width, Height), Composition, Row, Column, (HAxis, VAxis, NColums, Projection)
Width := <Number>          Height := <Number>
Composition := single | repeated | projection | ...
    the type of a visualization layout; repeated refers to small multiples

Row := RCItem[0..2]        Column:= RCItem[0..2] row and column items (as used in a Trellis plot)
RCItem:= Field<String> | FieldObject

HAxis := AxisItem          VAxis := AxisItem the design of an X and Y axis, respectively
AxisItem+ := (Domain, DomainColor, DomainDash, DomainWidth, DomainOpacity, Grid, GridColor, GridDash,
    GridWidth, GridOpacity, Offset)
NColumns := <Integer> the number of columns in a “repeated” composition
Projection+ := ProjectionType, ProjectionScale, ProjectionTranslate, ... the details of a map projection

FieldObject := Field<String>, (DataType, Scale, Sort, Aggregate, Bin) | Aggregate=count, (Scale)
    details about a data field encoded to a row/column/channel
DataType := nominal | ordinal | quantitative | temporal
Scale+ := (Domain, Range, Reverse<Boolean>, ...)
Domain := <Any>[]          Range := <Any>[] | ScaleName     ScaleName := <String> (e.g., “magma”)
Sort+ := ascending | descending | SortBy
SortBy := SortOrder, SortField<String> sort by a certain field
SortOrder := ascending | descending | <Any>[]
Aggregate+ := count | mean | max | median | ...
Bin+ := Maxbins | BinSteps | Nice | ...

Layer := LayerItem[]
LayerItem := Mark, (Text, Tooltip, Transform)

Mark := MarkType, ($MarkProperty)
MarkType := circle | point | bar | rect | ...
$MarkProperty ~ color, shape, size, stroke, ... mark properties or such channels
$MarkProperty := Value | FieldObject if “Value” is used, then it is a static property. Otherwise, it is an encoding channel.

Text := TextType, TextField, (Values, Anchor, Orient, TextItems, Tick, TextVisibility ...)
TextType := on-mark | on-axis | legend mark labels, axis labels, and legends, respectively
TextField := FieldName<String> the reference field of the text item (i.e., text element for each value of the “TextField”)
Values := <Any>[] a subset of elements of the TextField to show the labels
Anchor, Orient := start | end | middle | right-start | right-end | ....
    Anchor: the reference position to the corresponding mark/axis element; Orient: the reference position to the text element itself
TextItems := [Format, FontColor, Width, ...] each line of the text element
Tick := <Boolean> | (TickColor, TickWidth, ...)
    the design of the line segment between the text element and referred visual element.
TextVisibility := (External<Boolean>, Numbering<Boolean>, Position<top|bottom|left|right>)
    internalization/externalization of mark-labels; If Numbering = true, then reference numbers are shown on corresponding marks.

Tooltip := TooltipVisibility, TooltipFields
TooltipVisibility := on-mark | fixed | hidden the position of a tooltip
TooltipFields := [FieldName, Format] the information shown in a tooltip

Transform+ := TransformItem[] global/layer-specific data transformation
TransformItem := Filter | Aggregate, As | Bin, As | Compute, As, Op | ...

Interaction := InteractionItem[] global interactions
InteractionItem := ZoomPan | Context | Filter zoom+pan, brush-based context view, and interactive filter, respectively
```

```
ExVLSpec := (Name), Data, Layout, Layer, (Transform, Interaction, Title, NonData)

Title := (Width<Number>, Align), TitleItems
TitleItems := [(Name<String>), Text<String>, (Align, FontSize<Number>, FontWeight<100..900>, ...)]
    the design and content of each title element
Align := left | right | center

NonData := (NonDataVisibility, NonDataGlobalStyle,) NonDataItems
    annotation and emphasis elements that are not bound with data
NonDataVisibility := (External<Boolean>, Numbering<Boolean>, Position<top|bottom|left|right>)
NonDataGlobalStyle := (FontFamily<String>, FontWidth<100..900>, BoxStroke<Color>,
    BoxStrokeWidth<Number>, LineHeight<Number>, ...)
    global style of non-data items

NonDataItems := [NonDataType, (Name<String>), X<Number>, Y<Number>, DX<Number>, DY<Number>,
    Width<Number>, Height<Number>, Rotate<Number>, NonDataText, NonDataBox, NonDataMark]
    the appearance and contents of nondata items / X, Y: absolute position, DX, DY: relative position
NonDataType := text annotations | mark emphases
NonDataText := [Text<String>, (Align, Overflow, FontSize<Number>, FontWeight<100..900>,
    LineHeight<Number>, FontColor<Color>, Opacity<Number>)]
NonDataBox := (Padding<Number>, Fill<Color>, Stroke<Color>, StrokeWidth<Number>, Radius<Number>
    StrokeStyle)
StrokeStyle := solid | dashed | dotted | ...
NonDataMark := (Icon<String> | Image<URI> |t Shape), (Fill<Color>, Opacity<Number>, Stroke<Color>,
    StrokeOpacity<Number>, StrokeWidth<Number>, Radius<Number>)
Shape := circle | rect | rule | ...
```