In the theory for protobuf, there are a number of relations at play. Briefly, let's think about what each of the three relations would look like for CBOR:

- Value relation: Since each CBOR encoded value includes a header byte and argument (which in the case of an integer is actually the value of the integer, but for other types like text strings or byte strings would be the length of the string) before the value itself, we don't have the flexibility to slide values between types like we do in protobuf. Also, there is only one type of integer... which is were most of the work on protobuf is focused. This relation would likely be nearly an identity relation with tagged values providing the most likely departure from an identity relation.
- Compatible type relation: Like with the value relation, the one-to-one mapping from major types to base types in CDDL severely restricts the compatible type relation to be nearly an identity relation. However, once again tagged value and message types are likely to require some care.

Now, the above two points are *possibly* lies. The presence of controls in CDDL present an opportunity to add refinement types to the abstract value types, in which case the value and type relations would grow in size (and complexity) quite a bit. They would need to contain terms like $5 : \texttt{int} .\texttt{lt}\ 10 \prec 5 : \texttt{int} .\texttt{lt}\ 15$ and $\texttt{int} .\texttt{lt}\ 10 \propto \texttt{int} .\texttt{lt}\ 15$. Note that the value itself could never change, but the type can. The notation is a bit clumsy, in my opinion, but comes directly from CDDL.

In both protobuf and CDDL, the final compatibility relation is still a particular flavor of subtyping relation between message or map definitions. Each of the two relations have points of complexity. In Protobuf, everything is optional, allows for the addition of new fields which must obey the rules defined by the protobuf language. CBOR on the other hand has required fields by default, so the addition of new fields is limited to adding optional fields (and maybe pulling a field out of a wildcard repeated field?) while there is much more to think about with the refinement types that can be expressed in CDDL that protobuf lacks.