other things, the addition of leaf node B. Therefore, one can extract a temporal constraint from this relation:

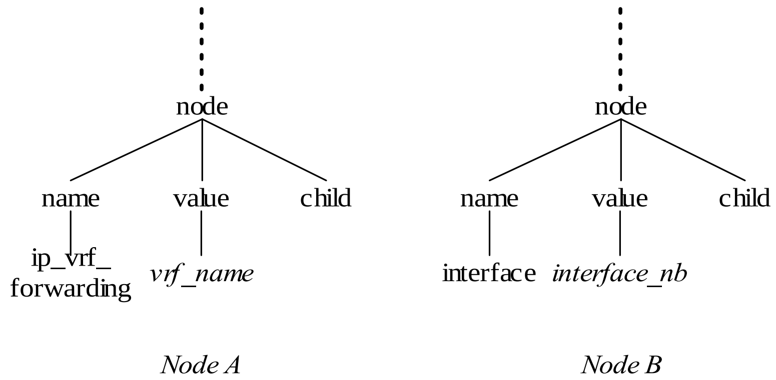

Fig. 1. Two configuration nodes that must be added for deploying a VPN

Temporal Constraint 3 *The node* `ip_vrf_forwarding` *cannot be added to a configuration tree before node* `interface/number` *has been set.*

Similar dependencies can be extracted for many other pairs of nodes among the 30 parameters involved, based on

- the semantic dependencies among the various components and parameters of the configuration;
- the spatial distribution of the configuration components and parameters;
- the choices of topology and technologies (protocols, device roles and types, vendor software, etc.).

These interdependencies imply a logically simultaneous configuration of the respective parameters on all these equipments. Since these equipments are spatially distributed and configuration operations can only be performed sequentially, this goal can only be achieved by "synchronizing" the configurations on different equipments by carefully setting up *validation points* during the configuration procedure.

The Netconf protocol [12] defines a simple mechanism for network device management. However, its transactional model, which includes a Validation capability, is device-centered, and does not provide a mechanism to ensure the consistence of the configuration that involves correlated configuration steps on multiple devices.

Netconf provides two phases of a successful configuration transaction during a service configuration procedure: preparation and commitment.

During preparation, the configurations are retrieved from the network devices. When all the configurations have been retrieved, the edition starts at service level. The validation at this stage ensures that the network configuration is consistent before the proposed modifications required by the service. To ensure the integrity of the configuration edition, the device configurations are locked, edited and subsequently unlocked. When the service edition has been successfully accomplished, the commitment starts. The validation at this stage ensures that the network configuration remains consistent after the respective modification of the network configurations.

Therefore, taken as is, Netconf does not provide any indication as to where and what to validate.

The previous examples have shown that many configuration operations must be done in a specific sequence, others must be performed together notwithstanding the order and others are mutually exclusive. Therefore, we need a clear temporal representation of the operations to be performed, which will describe all the temporal dependencies, indicate the possible procedural order of operation for various groups of configuration parameters on various devices and indicate the optimal temporal order and distribution of these operations.

3 A Theoretical Model

In this section, we present a theoretical study of the temporal issues described in section 2 by providing a theoretical model of the situation.

3.1 States and Transitions

Let S be a set of “states” representing a unit situation at a given time. In the case of network configuration, states are labelled trees described in section 2.

We call *transition* from a state s_1 to a state s_2 the structural modifications that transform s_1 into s_2. Formally, transitions can be defined as a subset of tuples $T \subseteq S \times S$; there exists a transition from s_1 to s_2 if and only if $(s_1, s_2) \in T$. The tuple (S, T) forms a directed graph G that we call a *transition diagram*.

In the case of the labelled trees we use for modelling device configurations, structural modifications are limited to addition of a labelled node to a leaf, or deletion of a leaf node in the tree. These modifications intuitively refer to addition, deletion or modification of a parameter in the configuration of a device. Therefore, it is possible that no transition exists in either way between two given states: this explains why T is only a subset of all possible pairs of states.

A *path* is a finite sequence of states $<s_1, ..., s_n>$ such that, for any s_i, s_{i+1}, there exists a $t \in T$ such that $t = (s_i, s_{i+1})$. The *distance* between two states, noted $\Delta(s_1,s_2)$ is the length of the shortest directed path linking them.

The configuration problem of the previous section becomes in this system the study of all paths that start from a given configuration, s_s, and end at a target configuration s_t. In addition, one might want to find the shortest of such paths.

However, this system, taken as is, is too general for any practical use. In particular, we must make sure that only solutions that progress towards the target are possible.We hence use path constraints to limit our study to sequences of states that have a meaning and are not degenerate.

A solution is to remove all tuples $(s_1,s_2) \in T$ such that $\Delta(s_1,s_t) < \Delta(s_2, s_t)$.

This condition makes sure that the parameters that are actually added are part of the solution, but not of the start state, and that parameters that are removed are part of the start state, but not part of the solution. Any other modification is out of the way of an acceptable solution. This distance restriction also has for effect of removing any loops in the paths.

3.2 Temporal Constraints

Now that G has been trimmed of any nonsensical states and paths, we can add further restrictions by imposing on the remaining transitions the semantic constraints related to the situation we are trying to model.

For example, in order to respect Temporal Constraint 3 in the case of the VPN deployment exposed in section 2.2, we must remove all transitions that lead to states where node A of Figure 1 is added while node B is not present.

More semantic constraints can be added to further trim the state graph from unwanted states and transitions. The remaining paths satisfy to all defined constraints. Intuitively, these so-called *acceptable* paths can be seen as a semantically desirable candidate solution for transforming the start state into the target final state.

3.3 Structuring Operations

In the minimal VPN example described previously, containing only two provider edge and two customer edge routers, the resulting state graph is