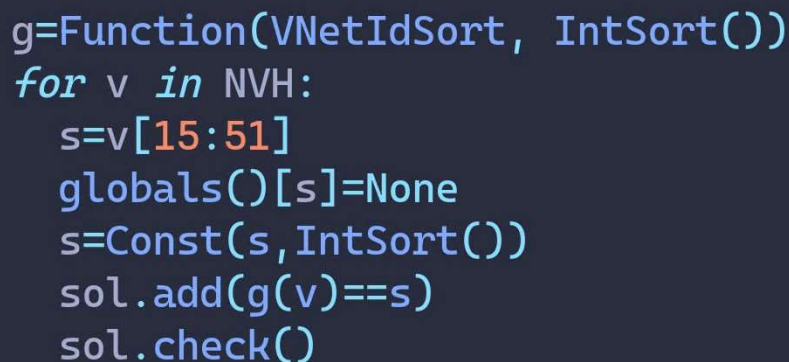


Automated reasoning in Azure and AWS

Version 1.1.1 (12 May 2022)

Proving the Hub & spoke pattern at scale
With Z3 SMT solver + EUF theory

Author: Christophe Parisel



```
g=Function(VNetIdSort, IntSort())  
for v in NVH:  
    s=v[15:51]  
    globals()[s]=None  
    s=Const(s,IntSort())  
    sol.add(g(v)==s)  
    sol.check()
```

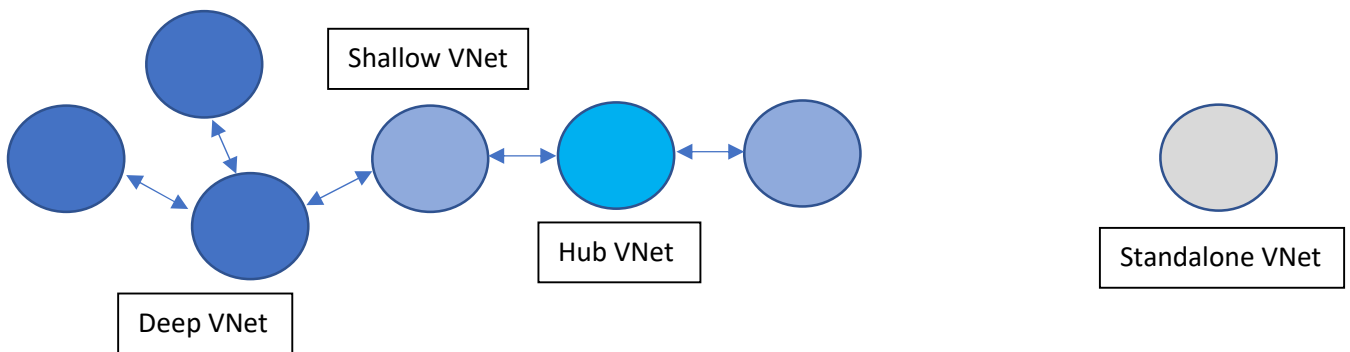
[linkedin.com/in/parisel](https://www.linkedin.com/in/parisel)

Assume we are given a random set V of VNet IDs and a non-empty pre-determined set H of subscription IDs.

Definitions:

- All members of V which belong to a subscription in H are called *Hub Vnets*,
- All members of V which are not Hub Vnets, but which are peered to a Hub Vnet are called *Shallow Vnets*,
- All other members of V with at least one peer are called *Deep Vnets*
- The remainder of V are called *Standalone Vnets*

Note that we make no assumptions about the peering of Hub Vnets: they may be fully meshed, grouped into independent small or large clusters...



Definition: V and H are said to *follow the Hub & Spoke pattern* if the three following properties are met:

- (CONNECTIVITY): there are no Standalone Vnets and there is at least one Hub Vnet,
- (SPOKE-TO-HUB): for any non-Hub Vnet v, there exists a chain of peerings between v and at least one Shallow VNet,
- (NO SPOKE-TO-SPOKE): all peers of any Deep Vnet v belong to the same subscription as v

Definitions:

- The subscription IDs of H are called *Hubs*.
- If V and H follow the Hub & Spoke pattern, the subscription IDs of V members which are not Hubs are called *Spokes*.

Let's take a deeper look at each of these 3 properties and see how we could give them a proof.

The CONNECTIVITY property

Proving this property is elementary.

The SPOKE TO HUB property

We start by examining the subscription IDs of all Vnets in V and sort them into two sets: Hub Vnets (HV) and non-Hub Vnets (NHV).

We then coalesce all NHV peers together: for each Vnet ID v1 in NHV, for each peer v2 of v1, we state the equality relationship **v1 == v2**.

By virtue of the equality relationship, NHV is now partitioned and all v in NHV are grouped into equivalence classes.

Let's introduce the uninterpreted function $h(\text{VNet ID}) \Rightarrow \text{VNet ID}$

For each Vnet ID v in the NHV partition, if v has at least one peer in H , we state the equality relationship:

$v == h(v)$

By this process, $h(v)$ is added to the equivalence class of v .

$h()$ has the property that if v is a Shallow Vnet, then $v == h(v)$. Note the converse is not always true since $h()$ is uninterpreted.

Also note that all the above can be calculated in $O(n)$, where n is the cardinality of NHV.

We can now prove the SPOKE-TO-HUB property in $O(n)$:

For each VNet ID v in NHV, we add the inequality $v \neq h(v)$ and check if it's satisfiable.

If it is not, we remove the inequality and carry on with the next member of NHV. If it is, then we have found an equivalence class of NHV which doesn't contain a $h(.)$ item. Put it another way, if $v \neq h(v)$ is satisfiable, there exists a group of Non-Hub Vnets represented by v which are not connected to any Shallow VNet or there are no Shallow Vnets in NHV.

At the end of the process, if we haven't managed to satisfy this inequality, then all Non-Hub Vnets are connected to at least one Shallow VNet, either directly or indirectly by transitivity of the equality relationship.

This proves the property at scale.

Before moving on to the NO SPOKE-TO-SPOKE property, we first introduce arithmetization.

ARITHMETIZATION

Let $u()$ be an uninterpreted function with the following signature: $\text{CustomSort} \otimes \text{Integer} \otimes \text{Integer} \Rightarrow \text{Integer}$

Definition: for j, i and for any x in CustomSort, $u(x, i, j)$ is arithmetized if $u(x, i, j) == \text{abs}(\text{hash}(\text{str}(x[i:j])))$ in Python.

Intuitively, **$u()$ is an oracle** that assigns a unique, deterministic integer to an uninterpreted symbol.

Let's introduce an arithmetized oracle $o()$ with signature: $\text{SubscriptionId} \Rightarrow \text{Integer}$

To that end, we set $i=0$ and $j=-1$ so we have $o(\text{subscriptionId}, 0, -1) = \text{abs}(\text{hash}(\text{str}(\text{subscriptionId}[0:-1])))$

Since $\text{string}[0:-1] == \text{string}$, we may drop i and j , and $o(x)$ simply returns the hash of the input symbol x .

NO SPOKE-TO-SPOKE property

For each subscription ID symbol s in NHV Vnets, we add the following oracular constraint to the Z3 model:

$$s == \text{abs}(\text{hash}(s)) \quad [\text{equation 0}]$$

So, each subscription ID symbol encountered in NHV is assigned a unique integer. Let's call **SID** the set of all such symbols.

We now introduce a variant of $o()$, the oracle $g()$ of type $VNetIdSort \otimes Integer \otimes Integer \Rightarrow Integer$

In the case of Azure, we set $i = 15$ and $j = 51$ so that for any $VNetId$ v , $g(v,i,j)$ is the subscription ID of v (it is shown in red below between characters 15 and 51 included):

```
/subscriptions/00000000-aaaa-bbbb-cccc-  
ffffffffffff/resourceGroups/RG/providers/Microsoft.Network/virtualNetworks/v,15,51)
```

This is not so straightforward in AWS since VPC ARNs are not of fixed length.

To get the account ID of a VPC, we may set $i = 22$ and $j = 35$ (corresponding to the account ID shown in green) when AWS region is us-east-1:

```
arn:aws:ec2:us-east-1:123456789012:vpc/vpc-1234567890abcdef0
```

The careful choice of indexes i and j ensures by [equation 0] that for any $VNetIdSort$ symbol x in NHV,

$$\exists s \in \text{SID} / g(x,15,51) = s \text{ (in Azure)} \quad \exists s \in \text{SID} / g(x,22,35) = s \text{ (in AWS)}$$

$$\text{So, for any two } VNetIdSort \text{ symbols } x \text{ and } y \text{ in NHV, } g(x,15,51) \neq g(y,15,51) \Rightarrow x \neq y \quad [\text{equation 1}]$$

All remains to be done is apply $g(.)$ sequentially to each $VNet$ Id symbol v in NHV's partition and see what happens in Azure (say):

$$g(v,15,51) == \text{str}(v[15:51]) \quad [\text{equation 2}]$$

If [equation 2] is not satisfiable for a given v , it means that $g(v,15,51)$ has already been assigned another value, so the NO SPOKE-TO-SPOKE property is disproved.

Why?

- For any w in the same class as v , we have $v == w$ by construction of the NHV partition
- During iteration over [equation 2], the subscriptionId of some w in the same equivalence class as v was already arithmetized as $\text{abs}(\text{hash}(\text{str}(w[15:51])))$ which is different from $\text{abs}(\text{hash}(\text{str}(v[15:51])))$ if v and w belong to distinct subscriptions
- If v and w belong to distinct subscriptions while belonging to the same equivalence class, we get $v == w$ and $g(v,15,51) \neq g(w,15,51)$ at the same time, which is not possible by [equation 1]

If [equation 2] is satisfiable for all $VNet$ ID in NHV, the NO SPOKE-TO-SPOKE property is proven.

Once again, all operations can be carried out in $O(n)$, ensuring scalability of the proof.

APPENDIX: Z3+python pseudo code

Spoke to hub property

```
from z3 import *
sol=Solver()
v1,v2=Consts('v1 v2',VNetIdSort)
for v1 in peers:
    globals()[v1]=None
    v1=Const(v1,VNetIdSort)
    for v2 in peers[v1]:
        globals()[v2]=None
        v2=Const(v2,VNetIdSort)
        sol.add(v1==v2)

h=Function(VNetIdSort, VNetIdSort)
for v in NHV:
    sol.push()
    sol.add(v!=h(v))
    sol.check()
    sol.pop()
```

No spoke-to-spoke property

(note that indexes i==15 and j==51 have been replaced by constants)

```
g=Function(VNetIdSort, IntSort())
for s in NHV's subscription IDs:
    globals()[s]=None
    s=Const(s,IntSort())
    subIdHash=abs(hash(str(s)))
    sol.add(s==subIdHash)

for v in NVH:
    s=v[15:51]
    globals()[s]=None
    s=Const(s,IntSort())
    sol.add(g(v)==s)
    sol.check()
```