Calculation of derivative properties in XMS-CASPT2

Peter John Cherry
Shiozaki group meeting
October 30th 2017

Calculation of matrix elements for many electron operators

- We need a generic approach for evaluating terms of the form

$$\sum_{\substack{ijkl\\wxyz}} \sum_{J} T^{\dagger}_{ijkl} g_{wxyz} q_{uv} \langle I | a_i a_j a^{\dagger}_k a^{\dagger}_l a^{\dagger}_w a^{\dagger}_x a_y a_z a^{\dagger}_u a_v | J \rangle c_J$$

- These terms can be decomposed into a sum of expressions of the following form:

$$\sum_{ijklmn}^{act} \sum_{J} \langle I | a^{\dagger}_i a^{\dagger}_j a^{\dagger}_k a_l a_m a_n | J \rangle c_J A_{ijklmn}$$

- Jae used code from SMITH3 to develop an efficient method of doing this in a non-relativistic framework.

- However, for cases involving spin-flipping interactions, we need to develop a method which takes advantage of spin symmetry.

Outline of method

Step 1 : Get operator information.

- Split input tensors up into blocks.
- Generate all possible contractions of these blocks.
- Identify which blocks/contracted blocks are equivalent.

Step 2 : Manipulate indexes.

- Rearrange indexes to normal order, applying range constraints.
- Put indexes into alternating order and group common indexes.

Step 3 : Generate computational proceedure.

- Merge all contributions from different brakets.
- Generate a task list to obtain contracted A-tensors.

Step 4 : Execute computational proceedure.

- Contract A-tensors with gamma matrices.

Step 5 : Loop over spin sectors and states.

- Repeat for all necessary spin sectors.

Outline of method

Step 1 : Get operator information.

- Split input tensors up into blocks.
- Generate all possible contractions of these blocks.
- Identify which blocks/contracted blocks are equivalent.

← Changes here

Step 2 : Manipulate indexes.

- Rearrange indexes to normal order, applying range constraints.
- Put indexes into alternating order and group common indexes.

← Changes here

Step 3 : Generate computational proceedure.

- Merge all contributions from different brakets.
- Generate a task list to obtain contracted A-tensors.

← Changes here

Step 4 : Execute computational proceedure.

- Contract A-tensors with gamma matrices.

← Working here

Step 5 : Loop over spin sectors and states.

- Repeat for all necessary spin sectors.

Generation of A-tensors

- Each A-tensor is defined by the original operators, $T^{\dagger}_{ijkl}$, g_{wxyz}, q_{uv} , and a set, X , of contraction indices and factors:

$$A_{abcd} = \sum_{t}^{x_t \in X} s_t \sum_{e_t f_t g_t h_t m_t n_t} \delta_{e_t f_t} \delta_{g_t h_t} \delta_{m_t n_t} T^{\dagger}_{ijkl} g_{wxyz} q_{uv}$$

$$X = \{(\{(a_1, b_1), (c_1, d_1), (e_1, f_1)\}, s_1), (\{(a_2, b_2), (c_2, d_2), (e_2, f_2)\}, s_2), ...\}$$

$$\{e_t, f_t, g_t, h_t, m_t, n_t\} \subset \{i, j, k, l, w, x, y, z, u, v\}$$

Contraction of γ^{I}_{ijklmn} and A_{ijklmn}

- Contraction of A-tensor with reduced density matrix (or similar) term:

$$\sum_{ijklmn}^{act} \sum_{J} \langle I | a_i^\dagger a_j a_k^\dagger a_l a_m^\dagger a_n | J \rangle c_J A_{ijklmn}$$

- The middle term is often too large to be stored in memory.
- Two possible methods may be used to calculate this term “on-the-fly” will be discussed at the end.

Contraction of γ^{I}_{ijklmn} and A_{ijklmn}

- Final step is to contract the A-tensor with the reduced density matrix or similar terms :

$$\sum_{ijklmn}^{act} \sum_{J} \langle I | a_i^\dagger a_j a_k^\dagger a_l a_m^\dagger a_n | J \rangle c_J A_{ijklmn}$$

- Aim to avoid storing the six index reduced density matrix derivative term in memory.

$$\sum_{ijkl} H^\dagger_{ijkl} \sum_{K} \langle I | ijk^\dagger | K \rangle \sum_{nop} \sum_{J} T_{lnop} \langle K | n^\dagger op | J \rangle c_J^N$$

Contraction of γ^{I}_{ijklmn} and A_{ijklmn}

- Another alternative is to use two three index terms

$$\sum_{J}\sum_{\substack{ijkl\\lnop}} H^{\dagger}_{ijkl} T_{lnop} \langle I|ijk^{\dagger}l^{\dagger}n^{\dagger}o|J\rangle c^{N}_{J}$$

$$= \sum_{ijkl} H^{\dagger}_{ijkl} \sum_{K} \langle I|ijk^{\dagger}|K\rangle \sum_{lnop}\sum_{J} T_{lnop} \langle K|l^{\dagger}n^{\dagger}o|J\rangle c^{N}_{J}$$

- Often there are only a few contributions to the A-tensor, which
- One possible method is to decompose it into a four index and two index term:

Generation of A-tensors

- Each A-tensor is defined by the original operators, $T^{\dagger}_{ijkl}$, g_{wxyz}, q_{uv} , and a set, X , of contraction indices and factors:

$$A_{abcd} = \sum_{x}^{x \in X} \sum_{efghmn} \delta_{ef}\delta_{gh}\delta_{mn}T^{\dagger}_{ijkl}g_{wxyz}q_{uv}$$

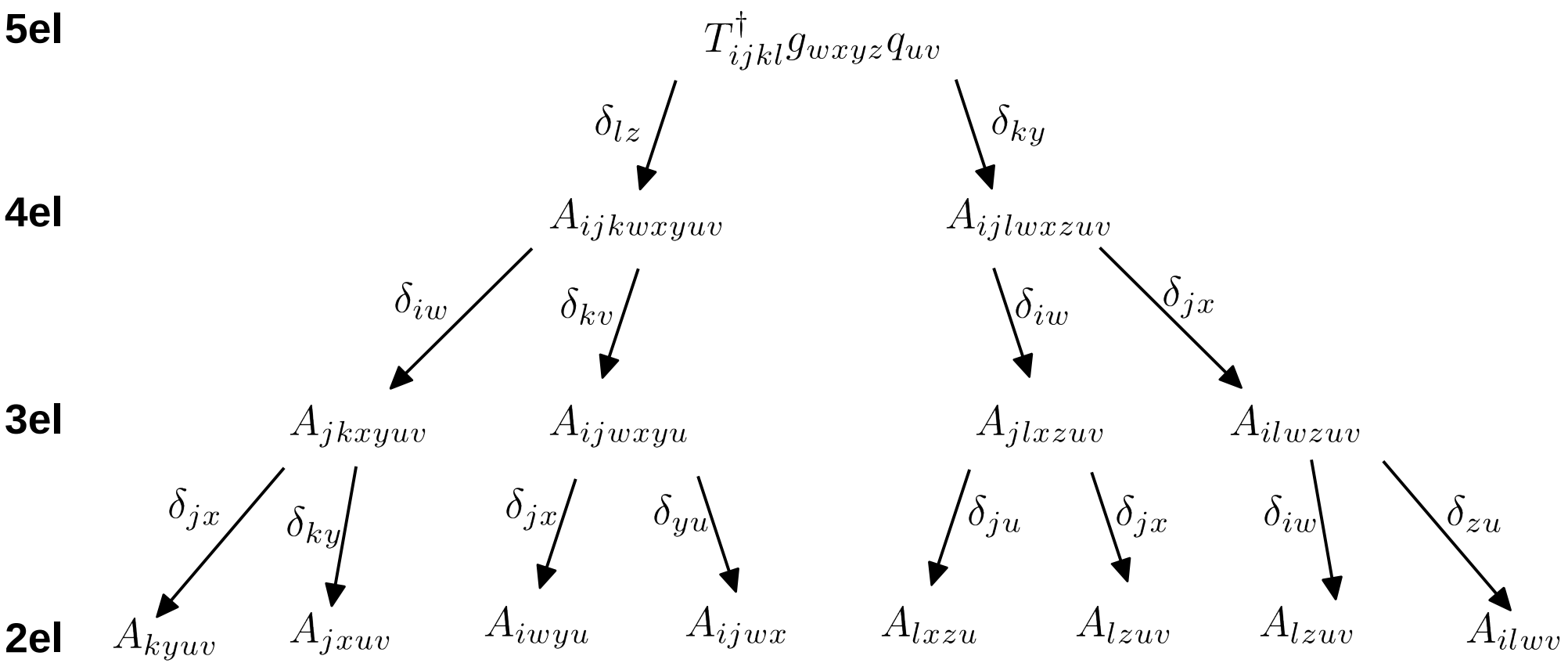
$$X = \{(\{(a_1, b_1), (c_1, d_1), (e_1, f_1)\}, s_1), (\{(a_2, b_2), (c_2, d_2), (e_2, f_2)\}, s_2), ...\}$$

- Each contribution, A^x , to the A-tensor is obtained by performing a recursive sequence of operations:

$$A^x_{abcd} = B(A^x_{abcdefgh}, (g, h)) =$$

$$= B(B(A^x_{abcdefgh}, (g, h)), (e, f)) = B(B(B(A^x_{abcdefgh}, (g, h)), (e, f)), (c, d))$$

Contraction tree



Building contraction tree

- The contraction tree is quite large; at each vertex connects to N_{el}^2 paths.
- This leads to $(N_{el}!)^2$ distinct paths (14400 for a five electron operator).
- Which paths we need are determined by how we reorder the expression in question:

$$\sum_{\substack{ijkl\\wxyz}} \sum_{J} T^{\dagger}_{ijkl} g_{wxyz} q_{uv} \langle I | a_i a_j a_k^{\dagger} a_l^{\dagger} a_w^{\dagger} a_x^{\dagger} a_y a_z a_u^{\dagger} a_v | J \rangle c_J$$

- Obtaining all A-tensors efficiently is not the only consideration; we must also consider the evaluation of final contraction:

$$\sum_{ijklmn}^{act} \sum_{J} \langle I | a_i^{\dagger} a_j a_k^{\dagger} a_l a_m^{\dagger} a_n | J \rangle c_J A_{ijklmn}$$

Generation of contraction task list

- Each A-tensor is defined by the original operators, $T^{\dagger}_{ijkl}$, g_{wxyz}, q_{uv} , and a set, X , of contraction indices and factors:

$$A_{abcd} = \sum_{x}^{x \in X} \sum_{efghmn} \delta_{ef}\delta_{gh}\delta_{mn} T^{\dagger}_{ijkl} g_{wxyz} q_{uv}$$

$$X = \{(\{(a_1, b_1), (c_1, d_1), (e_1, f_1)\}, s_1), (\{(a_2, b_2), (c_2, d_2), (e_2, f_2)\}, s_2), ...\}$$

- Each path from top to bottom corresponds to a member of X .
- Order of contractions is not important.
- Reorder the individual members of X to get the most efficient tree.

Initial reordering

$$T^{\dagger}_{ijkl} g_{wxyz} q_{uv} \langle I | a_i a_j a^{\dagger}_k a^{\dagger}_l a^{\dagger}_w a^{\dagger}_x a_y a_z a^{\dagger}_u a_v | J \rangle c_J$$

- Initial ordering is ill-suited to efficient computation. Reorder to normal order :

$$= T^{\dagger}_{ijkl} g_{wxyz} q_{uv} \langle I | a^{\dagger}_k a^{\dagger}_l a^{\dagger}_w a^{\dagger}_x a^{\dagger}_u a_i a_j a_y a_z a_v | J \rangle s_0$$

$$+ \sum_{\{\mu,\nu\}} T_{abcd} g_{efgh} q_{ij} \langle I | a^{\dagger}_q a^{\dagger}_r a^{\dagger}_u a^{\dagger}_v a_w a_x a_y a_z | J \rangle \delta_{\mu\nu} s_{\mu\nu}$$

$$+ \sum_{\{\mu,\nu\}} \sum_{\{\tau,\epsilon\}} T_{abcd} g_{efgh} q_{ij} \langle I | a^{\dagger}_q a^{\dagger}_r a^{\dagger}_u a_x a_y a_z | J \rangle \delta_{\mu\nu} \delta_{\tau\epsilon} s_{\mu\nu\tau\epsilon}$$

$$+ \sum_{\{\mu,\nu\}} \sum_{\{\tau,\epsilon\}} \sum_{\{\eta,\theta\}} T_{abcd} g_{efgh} q_{ij} \langle I | a^{\dagger}_q a^{\dagger}_r a^{\dagger}_u a_x a_y a_z | J \rangle \delta_{\mu\nu} \delta_{\tau\epsilon} \delta_{\eta\theta} s_{\mu\nu\tau\epsilon\eta\theta}$$

- Now all indexes in the BraKet must be active; this constraint greatly reduces the computational cost as typically $N_{act} \ll N_{virt}$ and $N_{act} \ll N_{core}$.

Six electron contribution

- The order of the indexes on the bra-ket and contractions still have the same indexes as the operators, T, g and q, but the ordering is changed:

$$\sum_{\{\mu,\nu\}} \sum_{\{\tau,\epsilon\}} T_{abcd} g_{efgh} q_{ij} \langle I | a_q^\dagger a_r^\dagger a_u^\dagger a_x a_y a_z | J \rangle \delta_{\mu\nu} \delta_{\tau\epsilon} s_{\mu\nu\tau\epsilon}$$

$$\{a, b, c, d, e, f, g, h, i, j\} = P\{\mu, \nu, \tau, \eta, u, v, w, x, y, z\}$$

- The ordering, sign and any constraints on the indexes are determined by the entire sequence of permutation operations necessary to get to that point.
- In terms of computational cost, which sequence of permutations is used is (almost) as important as the final index order.
- New terms due to reordering are always of equal or lower rank; apply reordering recursively from largest term to smallest.

Optimal ordering sequence

- By passing *through* certain orders, it is possible to eliminate terms and simplify the final expression.

Normal order

➔ Indexes in bra-ket must be active.

$$\langle I|a_i^\dagger a_j^\dagger a_k^\dagger a_l a_m a_n|J\rangle$$

Spin maximizing/minimizing order

➔ Reorder so as to maximally decrease or increase spin.

➔ May result in contractions (depending on the spin-sector).

$$\langle I|\alpha_i^\dagger \beta_j \beta_k^\dagger \alpha_l \beta_m^\dagger \alpha_n|J\rangle$$

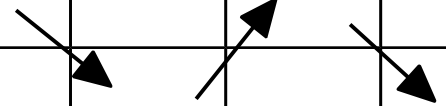
$$\alpha_i^\dagger = a_{i,\alpha}^\dagger$$

$$\beta_i^\dagger = a_{i,\beta}^\dagger$$

Spin constraints

Chose order so as to push the wavefunction out of the active space.

Spin sector	$\{\|J\rangle\}$	$\{\|K\rangle\}$	$\{\|L\rangle\}$	$\{\|I\rangle\}$
$[7\alpha 0\beta]$				
$[6\alpha 1\beta]$				
$[5\alpha 2\beta]$				
$[4\alpha 3\beta]$				
$[3\alpha 4\beta]$				
$[2\alpha 5\beta]$				
$[1\alpha 6\beta]$				
$[0\alpha 7\beta]$				



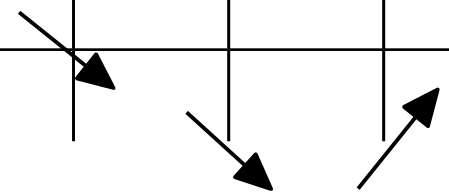
$$\sum_{KL} \langle I|\beta_k^\dagger \alpha_l|K\rangle\langle K|\alpha_i^\dagger \beta_j|L\rangle\langle L|\beta_m^\dagger \alpha_n|J\rangle =$$

$$= \begin{cases} 0 & \text{if} \quad i \neq n \quad \text{and} \quad i \neq k \\ \pm 1 & \text{otherwise} \end{cases}$$

Spin constraints

Chose order so as to push the wavefunction out of the active space.

Spin sector	$\{\|J\rangle\}$	$\{\|K\rangle\}$	$\{\|L\rangle\}$	$\{\|I\rangle\}$
$[7\alpha 0\beta]$				
$[6\alpha 1\beta]$				
$[5\alpha 2\beta]$				
$[4\alpha 3\beta]$				
$[3\alpha 4\beta]$				
$[2\alpha 5\beta]$				
$[1\alpha 6\beta]$				
$[0\alpha 7\beta]$				



$$\sum_{KL} \langle I|\alpha_i^\dagger \beta_j|K\rangle\langle K|\beta_k^\dagger \alpha_l|L\rangle\langle L|\beta_m^\dagger \alpha_n|J\rangle = 0$$

Optimal ordering

- By passing *through* certain orders, it is possible to eliminate terms and simplify the final expression.

<u>Grouped operator index order</u>

➔ May decompose expression into product of two sub expressions

$$\sum_{K} \sum_{\substack{ijkl \\ wxyz \\ uv}} T^{\dagger}_{ijkl} g_{wxyz} q_{uv} \langle I | a^{\dagger}_{k} a^{\dagger}_{l} a^{\dagger}_{w} a^{\dagger}_{x} a^{\dagger}_{u} a_{i} a_{j} a_{y} a_{z} a_{v} | J \rangle c_{J}$$

$$\rightarrow \sum_{J} \sum_{\substack{ijkl \\ wxyz}} T^{\dagger}_{ijkl} g_{wxyz} \langle I | a^{\dagger}_{k} a^{\dagger}_{l} a^{\dagger}_{w} a^{\dagger}_{x} a_{i} a_{j} a_{y} a_{z} | K \rangle \sum_{uv} \sum_{J} q_{uv} \langle K | a^{\dagger}_{u} a_{v} | J \rangle c_{J}$$

$$\rightarrow \sum_{J} \sum_{\substack{ijkl \\ wxyz}} T^{\dagger}_{ijkl} g_{wxyz} \langle I | a^{\dagger}_{k} a^{\dagger}_{l} a^{\dagger}_{w} a^{\dagger}_{x} a_{i} a_{j} a_{y} a_{z} | K \rangle \tilde{c}_{K}$$

Optimal ordering

- By passing *through* certain orders, it is possible to eliminate terms and simplify the final expression.
- Final order should be determined from symmetry constraints, and the need to minimize the number of vertices high in the contraction tree.

Application of block symmetry

Example : CASPT2 perturbation coefficients $T_{(ij),(kl)}$.

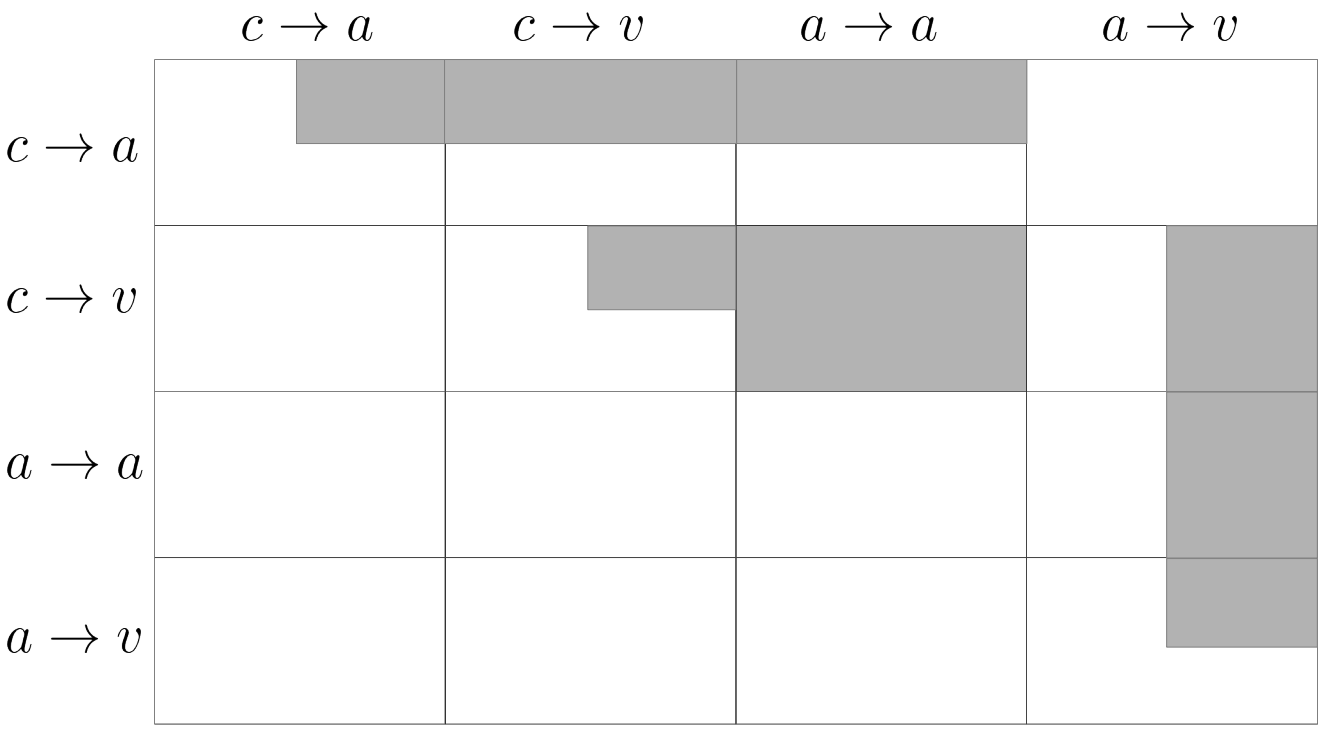
	$c \to a$	$c \to v$	$a \to a$	$a \to v$
$c \to a$				
$c \to v$				
$a \to a$				
$a \to v$				

Possible transitions

$$closed \to active$$
$$closed \to virtual$$
$$active \to active$$
$$active \to virtual$$

Application of symmetry

Example : CASPT2 perturbation coefficients $T_{(ij),(kl)}$.



Possible transitions

$closed \to active$
$closed \to virtual$
$active \to active$
$active \to virtual$

Contraction of tensor blocks

Contract over all possible pairs of indexes, e.g.,

$$\hat{H}\hat{T} \rightarrow \sum_{wxyz}\sum_{ijkl} H_{wxyz} a_w^\dagger a_x^\dagger a_y a_z T_{ijkl} a_i^\dagger a_j^\dagger a_k a_l$$

$$A^{HT,st}_{abcdef} = \sum_{wxyz}\sum_{ijkl} H_{wxyz} T_{ijkl} \delta_{st}$$

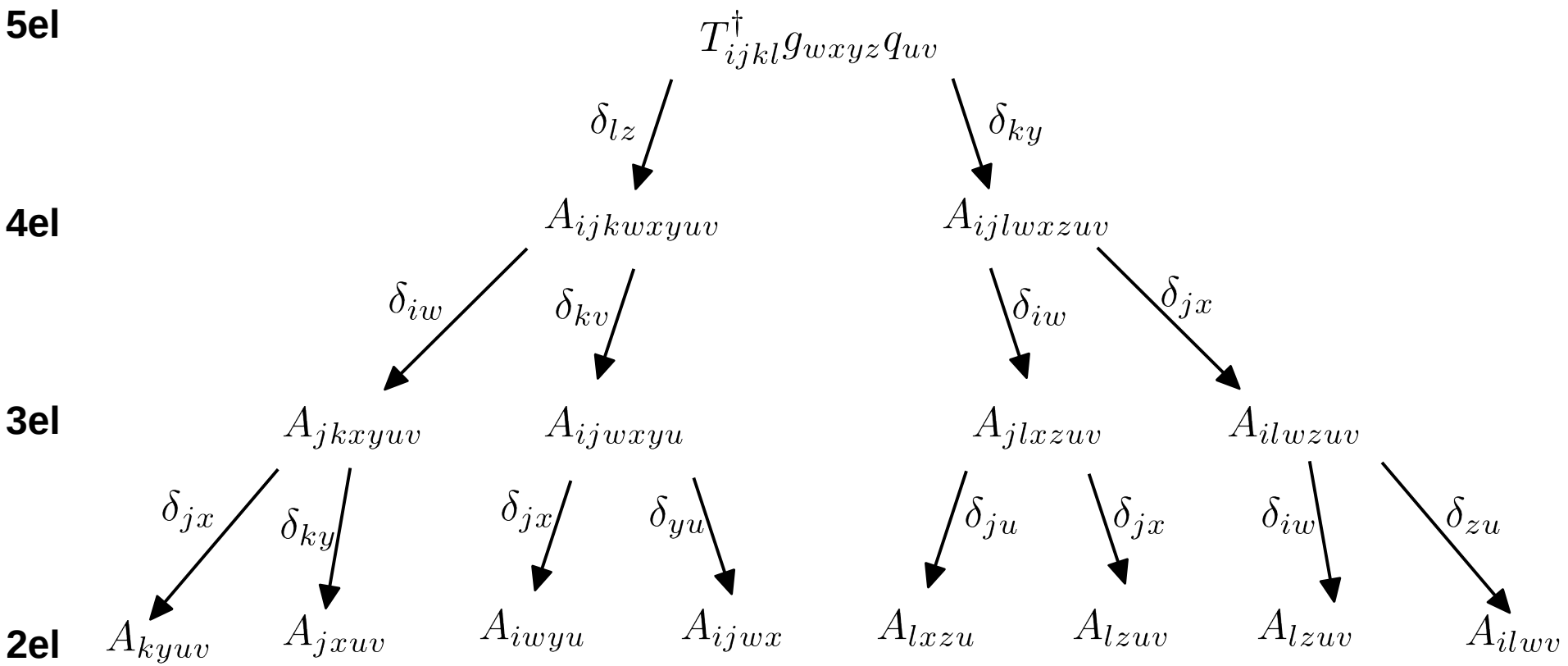
$$s := w, x, i, \text{ or } j$$
$$t := y, z, k \text{ or } l$$

Use the block range constraints to rule out possible contractions, e.g.

$$A^{HT,st}_{abcdef} = \begin{cases} A^{HT,st}_{abcdef} & \text{if} \quad rng(s) = rng(t) \\ 0 & \text{otherwise} \end{cases}$$

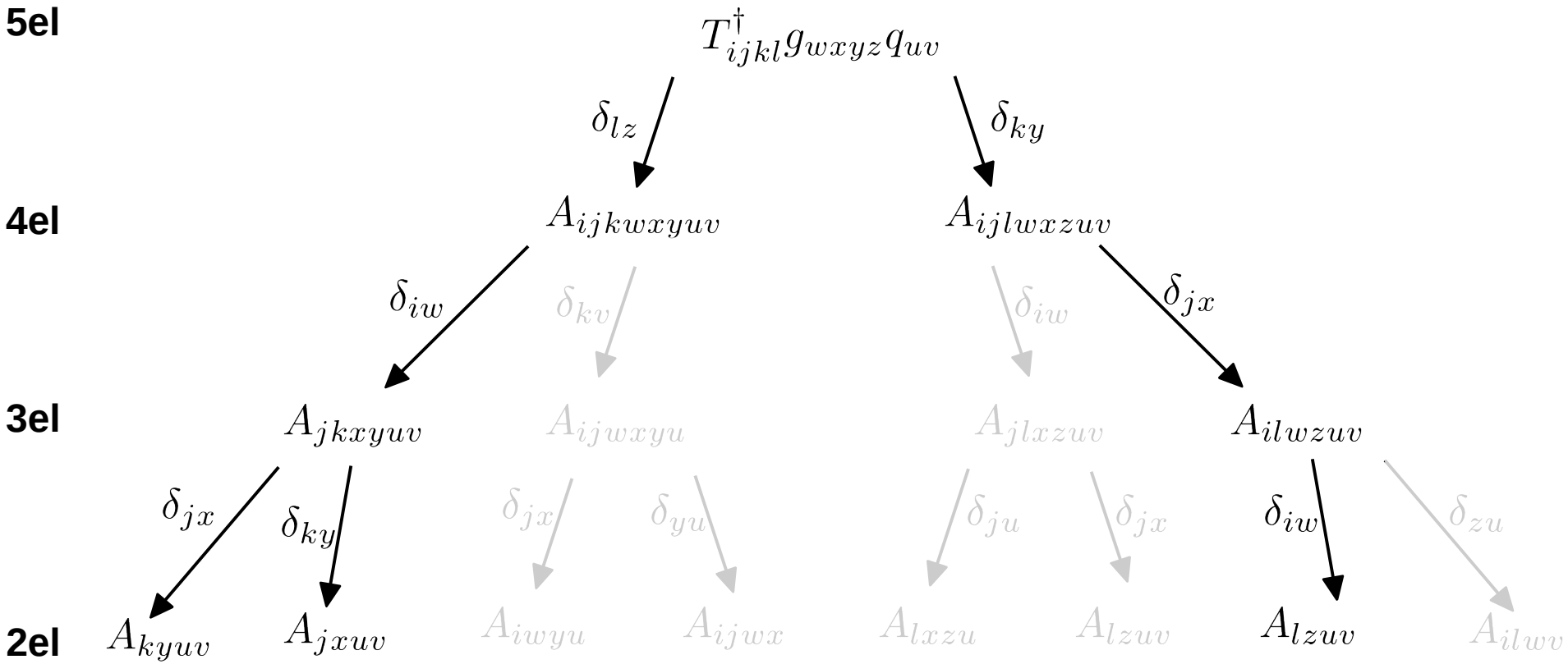
Contraction list is different for *every* range block.

Contraction tree



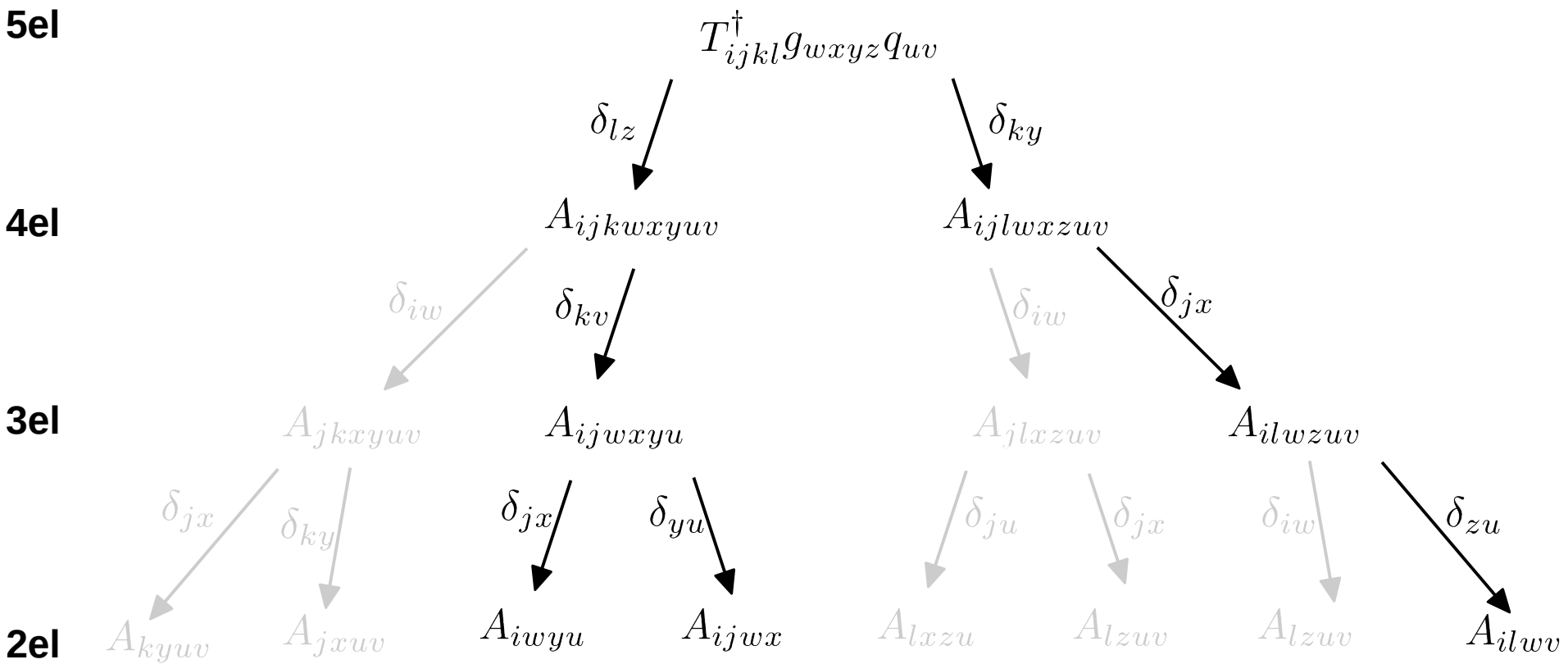
Block specific contraction trees

- Which paths contribute is depends on the block(s) under consideration.



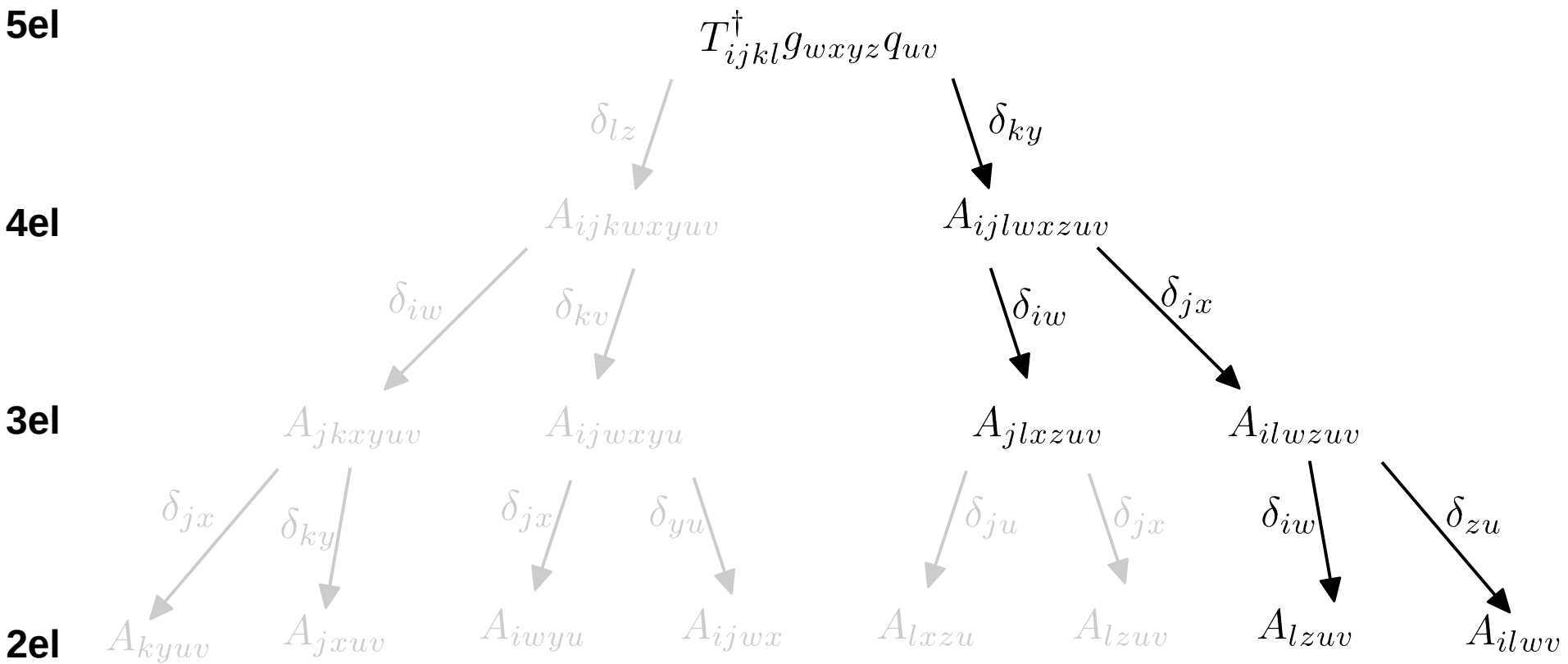
Block specific contraction trees

- Which paths contribute is depends on the block(s) under consideration.

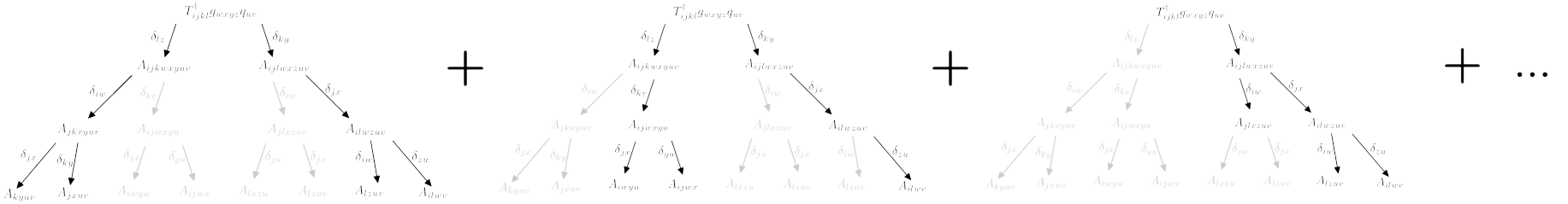


Block specific contraction trees

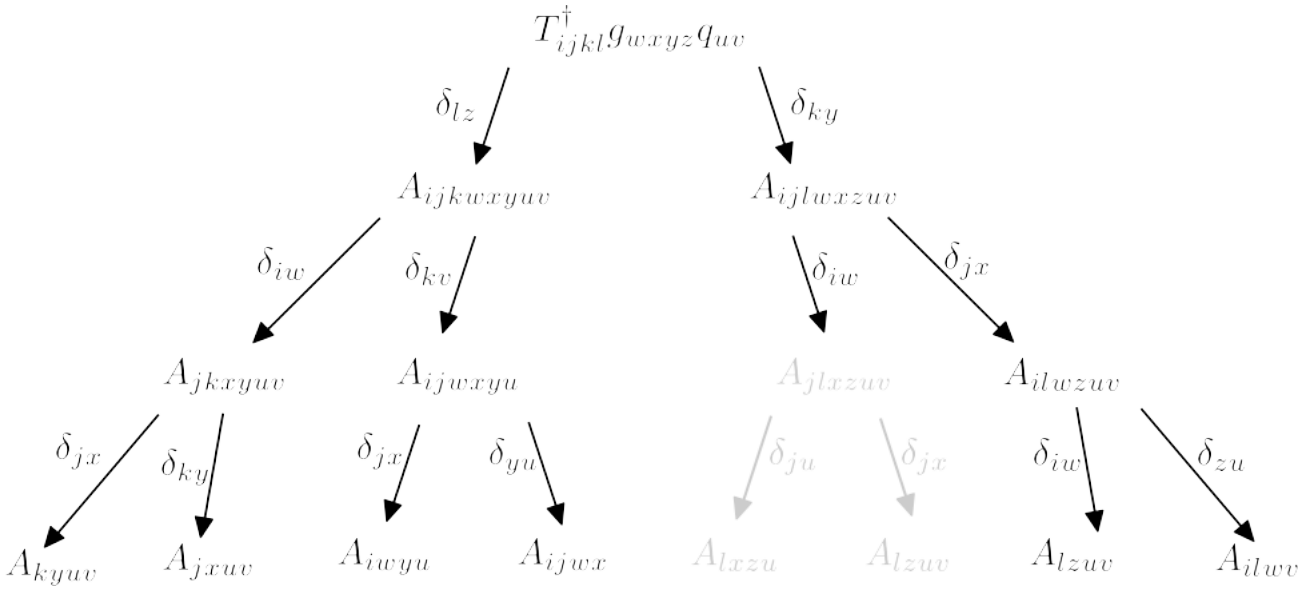
- Which paths contribute is depends on the block(s) under consideration.



Merge block specific contraction trees



=



Application of block symmetry

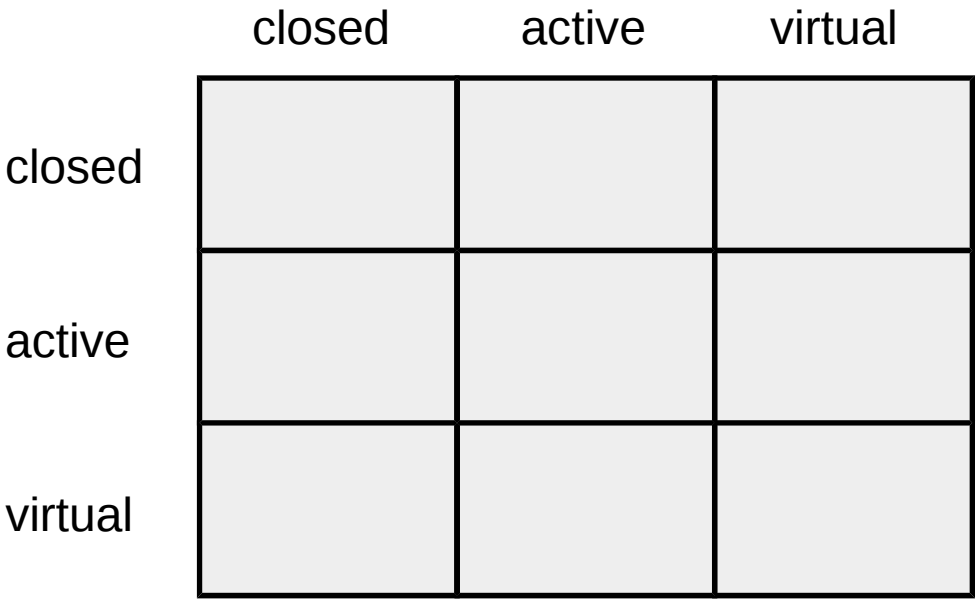
- The tree for each block is generated seperately.
- Each vertex on the tree is an object which contains information about itself and it’s position in the tree.
  - All requirements (vertexes higher up in the tree).
  - All dependents (vertexes lower in the tree).
- Due to the block symmetry several vertexes are equivalent, which can lead to redundancies.
- To avoid redundancies, block symmetry should be accouted for not just in the final contraction, but also in the construction of the tree.
- Currently, a very basic approach based on index substitution and standardized ordering is used.

Application of block symmetry

- The tree for each block is generated seperately.
- Each vertex on the tree is an object which contains information about itself and it’s position in the tree.
  - All requirements (vertexes higher up in the tree).
  - All dependents (vertexes lower in the tree).
- Due to the block symmetry several vertexes are equivalent, which can lead to redundancies.
- To avoid redundancies, block symmetry should be accouted for not just in the final contraction, but also in the construction of the tree.
- Currently, a very basic approach based on index substitution and standardized ordering is used.

Application of block symmetry

- Currently, a very basic approach based on index substitution is used.
- All indexes are named, and the program reorders them so they always occur in a certain order.



Closed-virtual block

Id names : $\{x_0, x_1\}$
Ranges : $\{clo, vir\}$
CA ops : $\{a^\dagger, a\}$

Virtual-closed block

Id names : $\{x_1, x_0\}$
Ranges : $\{clo, vir\}$
CA ops : $\{a^\dagger, a\}$

Contraction of γ^{I}_{ijklmn} and A_{ijklmn}

- Final step is to contract the A-tensor with the reduced density matrix or similar terms :

$$\sum_{ijklmn}^{act} \sum_{J} \langle I | a_i^\dagger a_j a_k^\dagger a_l a_m^\dagger a_n | J \rangle c_J A_{ijklmn}$$

- Aim to avoid storing the six index reduced density matrix derivative term in memory.

- One possible approach is to consider contributions to the A-tensor individually, enabling us to avoid evaluation of any density matrices with more than three indices :

$$\sum_{J} \sum_{ijkl} H^\dagger_{ijkl} T_{lnop} \langle I | ijk^\dagger l^\dagger n^\dagger o | J \rangle c_J^N$$

$$= \sum_{ijkl} H^\dagger_{ijkl} \sum_{K}^{lnop} \langle I | ijk^\dagger | K \rangle \sum_{lnop} \sum_{J} T_{lnop} \langle K | l^\dagger n^\dagger o | J \rangle c_J^N$$

Contraction of γ^I_{ijklmn} and A_{ijklmn}

- May be evaluated by the following sequence:

$$\sum_{ijkp} H^\dagger_{ijkp} \sum_K \langle I|ijk^\dagger|K\rangle \sum_{lno} \sum_J T_{lnop} \langle K|l^\dagger n^\dagger o|J\rangle c^N_J$$

(1) $$\sum_{LJ}^{N_{det}} \langle K|l^\dagger|L\rangle \langle L|n^\dagger o|J\rangle c^N_J = \sigma^K_{lno} \qquad N^3_{act} N_{det(J)}$$

(2) $$\sum_{nop}^{N_{act}} T_{lnop} \sigma^K_{lno} = \tilde{x}^K_p \qquad N^3_{act} N_{rng(p)} N_{det(K)}$$

(3) $$\sum_{l}^{N_{rng(p)}} H^\dagger_{ijkp} \tilde{x}^K_p = \tilde{w}^K_{ijk} \qquad N^3_{act} N_{rng(p)} N_{det(K)}$$

(4) $$\sum_{ijk}^{N_{act}} \sum_{K}^{N_{det}} \langle I|ijk^\dagger|K\rangle \tilde{w}^K_{ijk} = \tilde{c}^I \qquad N^3_{act} N_{rng(p)} N_{det(K)}$$

Contraction of γ^{I}_{ijklmn} and A_{ijklmn}

$$\sum_{ijkp} H^{\dagger}_{ijkp} \sum_{K} \langle I|ijk^{\dagger}|K\rangle \sum_{lno} \sum_{J} T_{lnop} \langle K|l^{\dagger}n^{\dagger}o|J\rangle c^{N}_{J}$$

- Avoid storing the rdm derivatives and A-tensor at the cost of recalculating the third order rdm multiple times.
- Key advantage is that rdm derivatives (or similar) and large A-tensor terms are not stored.
- Only the ci-vector and relevant tensor blocks are required; reduces communication.
- No reordering of indexes of large tensors.
- Becomes very costly if there are multiple contractions between different tensors, due to the occurence of N^{2}_{virt} terms.